

**GEORGIA DOT RESEARCH PROJECT 17-15**

FINAL REPORT

**Extended Field Testing and Enhancement of a Portable  
Pedestrian and Cyclist Detection System**

Sponsored By



**Office of Performance-based Management and Research**

600 West Peachtree Street NW | Atlanta, GA 30308

1. Report No.: FHWA-GA-20-1715		2. Government Accession No.: N/A		3. Recipient's Catalog No.: N/A	
4. Title and Subtitle: Extended Field Testing and Enhancement of a Portable Pedestrian and Cyclist Detection System			5. Report Date: March 2020		
			6. Performing Organization Code: D8310.0.0.0.0		
7. Author(s): Colin Usher, Wayne Daley			8. Performing Organ. Report No.: 17-15		
9. Performing Organization Name and Address: Georgia Tech Research Institute 640 Strong Street Atlanta, GA 30318			10. Work Unit No.: N/A		
			11. Contract or Grant No.: PI#0015710		
12. Sponsoring Agency Name and Address: Georgia Department of Transportation Office of Performance-based Management and Research 600 West Peachtree Street NW Atlanta, GA 30308			13. Type of Report and Period Covered: Final; September 2017-March 2020		
			14. Sponsoring Agency Code: N/A		
15. Supplementary Notes: Prepared in cooperation with the U.S. Department of Transportation, Federal Highway Administration					
16. Abstract: The objective of this project was to enhance the capabilities of the pedestrian tracking system by adding support for detection and tracking of cyclists and vehicles, in addition to generating a set of tools to aid in analysis. Secondary tasks explored methods of enhancing the system capability for added value. The system analyzes the collected video data and automatically identifies and characterizes the number of pedestrians, vehicles, and cyclists and their behavior at midblock and intersection locations (specifically, their crossing locations and frequency). The system is designed to be portable to support data collection in multiple locations. Key conclusions from this study are:  1) A detection and tracking software system capable of creating and logging trajectories for pedestrians, vehicles, and cyclists was implemented and tested. 2) Field testing of detection algorithms yielded accuracy between 90% and 95%, but count accuracy suffered at 53%. 3) A set of powerful analysis tools was developed to allow in-depth analysis of results.					
17. Key Words:  Pedestrian, Automated, Tracking, Planning			18. Distribution Statement:  No Restrictions		
19. Security Classification (of this report): Unclassified		20. Security Classification (of this page): Unclassified		21. Number of Pages: 132	22. Price: Free

GDOT Research Project 17-15

Final Report

EXTENDED FIELD TESTING AND ENHANCEMENT OF A PORTABLE PEDESTRIAN  
AND CYCLIST DETECTION SYSTEM

By

**Colin Usher**

Senior Research Scientist

**Wayne Daley**

Principal Research Engineer

Georgia Tech Research Institute

Contract with

Georgia Department of Transportation

In cooperation with

U.S. Department of Transportation  
Federal Highway Administration

March 2020

The contents of this report reflect the views of the author(s) who is (are) responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Georgia Department of Transportation or the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

# Table of Contents

Executive Summary .....	1
Introduction .....	3
Objectives/Tasks .....	5
1) Assist GDOT in processing and generating reports .....	7
2) Improve reporting system capabilities.....	10
3) Explore data processing with different video sources .....	40
4) Evaluate the ability to automatically perform incident detection or generate a “danger/safety” metric based on vehicle and pedestrian interactions .....	45
5) Improve trailer hardware system.....	60
6) Perform training and demonstration of the system for GDOT personnel.....	61
Anticipated Impact .....	61
Conclusions and Recommendations.....	62
References .....	63
Appendix 1 – Software Manuals .....	65
Appendix 2 - Camera Calibration Technical Document .....	91
Appendix 3 – Camera Intrinsic Parameters Technical Document.....	101
Appendix 4 – Literature Review Summary .....	108
Appendix 5 – Technical Document for Conflict Analysis .....	121

## List of Tables

Table 1 - Cyclist Detection and Tracking Accuracy .....	23
Table 2 - Short Range Position Estimation Results.....	27
Table 3 - Long Range Position Estimation Results.....	27
Table 4 - Tracker Database Query Optimization Results.....	33
Table 5 – Pedestrian Accuracy Results .....	37
Table 6 - Vehicle Detection and Tracking Accuracy .....	39
Table 7 - Conflict Detection Sample Data .....	45
Table 8 - Conflict Table .....	47
Table 9 - Conflict Table for Site 1.....	51
Table 10 - Conflict Table for Site 2.....	52
Table 11 - Conflict Table for Site 3.....	54
Table 12 – Normalized Conflict Results .....	57
Table 13 - Conflict Analysis Validation.....	59
Table 14 - False Positive Analysis for Conflict Reports .....	60
Table 15 - Detector and Tracker Accuracy .....	62

## List of Figures

Figure 1 - Site 1 .....	8
Figure 2 – Site 2 .....	8
Figure 3 – Site 3 .....	9
Figure 4 - Average Inference Time Comparison for multiple detectors .....	11
Figure 5 - MultiMon Processing Pipeline .....	14
Figure 6 - Current Database Tables and Fields .....	15
Figure 7 - Screenshot of Video Playback Tool.....	16
Figure 8 – Sample Image Generated by Report Generation Tool Containing Trajectory and Pedestrian .....	17
Figure 9 - Before and After User Interface Screenshots .....	18
Figure 10 - Windows User Interface Operation .....	19
Figure 11 - New User Interface Screenshot.....	19
Figure 12 - Sample System Capture of a Cyclist.....	20
Figure 13 - Sample Image from Campus Data Collection Containing a Cyclist.....	21
Figure 14 - Cyclist Track Results .....	21
Figure 15 - Cyclist Tracks from Cyclists Using Vehicle Lanes .....	22
Figure 16 - Cyclists Detected as Both Cyclists and Pedestrians .....	22
Figure 17 - Predicted Zoom vs. Actual Using Camera Model .....	25
Figure 18 - Calibration Target for Camera Extrinsic Parameters.....	25
Figure 19 - Camera Extrinsic target .....	26
Figure 20 - Campus Testing of GPS Location Capability .....	28
Figure 21 - New Database Structure .....	29
Figure 22 - Time-Test Results from Database Queries .....	30
Figure 23 - Pedestrian Tracker Accuracy Negatively Affected by High Volumes .....	31

Figure 24 - Inference Time for Detection and Tracking .....	32
Figure 25 - GPU Memory Usage.....	32
Figure 26 - Hardware Scaling for Simultaneous Video Processing .....	34
Figure 27 - Video Data Capture at Intersection on GT Campus .....	35
Figure 28 - Sample of Positive (Top) and False (Bottom) Detections .....	36
Figure 29 – Intersection and Sidewalk with Pedestrian Tracks.....	37
Figure 30 - Vehicle Trajectories.....	38
Figure 31 - Pedestrian Activity at Same Location as Vehicles in Above Image.....	38
Figure 32 - Vehicle Tracker Sample Screenshot.....	39
Figure 33 - Intersection with Heavy Pedestrian Activity .....	41
Figure 34 - Vehicle Tracks Illustrating Tracker Errors Due to Configuration .....	41
Figure 35- Fixed Tracker Errors Due to Configuration Settings .....	42
Figure 36 - Vehicle Occupants Counted as Pedestrians Before and After .....	42
Figure 37 - Sample Detector Output .....	43
Figure 38 - Sample Detector Output .....	43
Figure 39 - Sample Detector Output .....	44
Figure 40 - Sample Detector Output .....	44
Figure 41 - Conflict Plot.....	46
Figure 42 - Conflict Validation .....	47
Figure 43 - Conflict Plot.....	47
Figure 44 - False Conflict Sample.....	48
Figure 45 - Site 1 along Buford Highway .....	49
Figure 46 - Site 2, Intersection of J.E Lowery and J.E. Boone .....	49
Figure 47 - Site 3, Intersection of J.E. Lowery and M.L.K.....	50
Figure 48 - Site 4 Located at 10th and Myrtle.....	50
Figure 49 - Conflict Plot for Site 1, Camera 4.....	52

Figure 50 - Conflict Plot for Site 2, Camera 4.....	53
Figure 51 - Conflict Plot, Site 2 Camera 2, Construction Zone .....	54
Figure 52 - Raw Image and Associated Conflict Plot for Site 3, Camera 4 .....	55
Figure 53 - Conflict Distribution by Type at Site 4.....	56
Figure 54 - Raw Image and Associated Conflict Plot for Site 4 .....	56



## **Executive Summary**

Researchers at the Georgia Tech Research Institute (GTRI) developed and tested a portable pedestrian detection system for characterizing pedestrian behavior at both midblock and intersection locations throughout the state of Georgia. The system consists of a trailer with four high-resolution pan-tilt cameras and a suite of software for collecting video data, processing the video to detect pedestrians and track trajectories, and generate reports based on the tracked results.

Under previous efforts, three software tools were developed: a program to enable video recording and saving; a video analysis tool to allow preparation of video for processing and generating pedestrian trajectory data that is logged to a local database; and a reporting tool that allows for generation of reports and analysis of pedestrian crossing behavior for particular time periods. Results from the report generation tool include a visualization of the pedestrian crossings for the user-defined time period to allow for analysis.

Several new tools were developed under the scope of this project to assist in higher-value analysis and decision making. These tools consisted of a software program to view pedestrian crossings in video using data generated by the report generation tool and a tool to analyze trajectory data and perform incident detection with the goal of developing a “safety” metric. The video playback tool was also modified to allow viewing of potential conflicts in video using data generated from the conflict analysis tool.

In addition to detecting and tracking pedestrians, the system was improved to allow detection and tracking of vehicles and cyclists. This required a major overhaul and development of a completely new detection and tracking system. This new system not only improved capabilities of the system, it also significantly improved processing time and reduced detection errors.

Average detection accuracy for pedestrians, vehicles, and cyclists hovers at approximately 95%. However, related counts as generated by the report generation tool have an accuracy of approximately 53%. The potential to detect and place trajectories in a geo-spatial frame was also explored. The team demonstrated the ability to place trajectories in a geo-spatial frame when the geography is very flat and planar, but identified issues with complex scenes.

The research team identified areas of potential improvement to add value to the data collected as well as to improve tracker performance. These include but are not limited to: advanced report generation algorithms that allow more robust combining of single pedestrians that due to busy scenes or occlusions are tracked and logged as multiple pedestrians; and development of methods to map undulating geographies to two-dimensional video data for estimating physical location in geo-spatial coordinates for complex scenes.

The result of the project is an operational field-tested prototype system that can be utilized by the GDOT to perform detailed analysis of pedestrian, cyclist, and vehicle behavior and interactions in complex scenes. The trailer-mounted video system allows quick deployment and ability to capture video from locations that may not have current video coverage. The software also allows for processing of video streams from several different sources, in addition to the trailer-mounted recording system. The suite of tools allows for in-depth analysis and validation of the system operation.

## **Introduction**

Pedestrians involved in roadway accidents account for nearly 12% of all traffic fatalities and 59,000 injuries each year [1]. Most injuries occur when pedestrians attempt to cross roads, and there have been noted differences in accident rates midblock vs. at intersections [2], [3].

Additionally, vehicle-pedestrian collisions at midblock also tend to be more deadly for pedestrians [2]. This is of significant concern to the Georgia Department of Transportation (GDOT) which is being proactive in exploring various approaches to increasing pedestrian safety [4].

Pedestrian behavior at midblock crossings in the metro Atlanta area is largely unknown. This leads to a lack of information to guide the proper design of lane markings and traffic signals to enhance pedestrian safety. The problem stems from observations that "...When convenient and manageable crossing points are not identified, most pedestrians cross at random, unpredictable locations. In making random crossings, they create confusion and add risk to themselves and drivers..." [5]. Techniques for enhancing pedestrian safety are well known [1], [6]; however, what is lacking are data to support the choice and location of countermeasures at a local level.

The current practice of collecting data on pedestrian behavior is a time-consuming manual process that is prone to error as well as limited in function. To address this, the Georgia Tech Research Institute (GTRI) developed and delivered to the Georgia Department of Transportation (GDOT) a state-of-the-art prototype automated system that uses video feeds to count pedestrians and cyclists crossing specifically at midblock locations on roads. Results from field testing indicate that the system has an average pedestrian detection rate greater than 90% with very low false positives (<5%) in midblock locations. A suite of software tools allows for processing of the video along with visualization and reporting of pedestrian counts and frequencies.

Operating the system in locations other than a midblock (such as in an intersection) had less than desirable operation of the detection system. In addition, through development and demonstration of the system, both the research team and GDOT personnel identified several areas where the system could be improved to potentially provide significantly higher value data.

The objective of this work was to support the GDOT in an extensive field test of the aforementioned system and to improve its current capabilities from both a user interface and a processing perspective. GTRI processed and generated several reports with the system, all while improving the system algorithms and operation. The system was modified and tested to operate in non-midblock locations such as intersections and other pedestrian heavy locations. During this time, additional tools for data analysis and report generation were also developed and tested with input from the GDOT.

## Objectives/Tasks

The objective of this project was to perform field testing and enhance the capabilities of a portable automated system to collect continuous video data on pedestrian and cyclist behavior at various locations throughout the metro Atlanta area. The previous system analyzes collected video data to automatically identify and characterize the number of pedestrians and their behavior at midblock locations (specifically, their crossing locations and frequency). Enhancements included modifying the pedestrian tracking algorithms to allow for operation in places other than midblock locations, exploring the ability to log pedestrian trajectories in a GIS system using GPS coordinates, and adding vehicle tracking capabilities to the system.

The following tasks were executed by GTRI to achieve the specific objectives:

1. Assist GDOT in processing and generating reports.
2. Improve reporting system capabilities.
  - a. Add ability to view pedestrian crossings in video immediately using detection database.
  - b. Refine the user interface based on feedback provided from GDOT.
  - c. Explore the ability and accuracy for the system to identify cyclists and pedestrians separately (current system identifies, but does not specify).
  - d. Explore the ability to determine GPS coordinates of crossings to allow data to be plotted in a GIS system.
  - e. Make further improvements to system processing time.
  - f. Improve and characterize system ability to track pedestrians in intersections.
  - g. Add and evaluate vehicle tracking abilities.
3. Explore data processing with different video sources.

4. Evaluate the ability to automatically perform incident detection or generate a “danger/safety” metric based on vehicle and pedestrian interactions.
5. Improve trailer hardware system.
6. Perform training and demonstration of the system for GDOT personnel.

The following sections of this report describe in detail the work completed for each of the tasks listed directly above.

## **1) Assist GDOT in processing and generating reports**

The original intent of this task was to operate the video capture trailer system with GDOT personnel to generate reports utilizing the pedestrian tracking software. Early in the project, the State Bicycle and Pedestrian Engineer moved into another position at the GDOT. When the replacement arrived several months into the project, work was well underway implementing the new tracker capabilities discussed in this report. Due to the timing of this, it was determined that efforts would be best spent operating the system and testing/validating the new tracker performance; therefore, no official reports were generated at the behest of the GDOT.

However, the trailer-mounted recording system was deployed at several locations around Georgia Tech and the City of Atlanta during development of the project. Sites included routes along Techway on campus, as well as several locations along pedestrian and vehicle routes. In addition, as part of task 2f, video from several GDOT cameras located around downtown Atlanta was processed. Results from these analyses are presented in the associated sections.

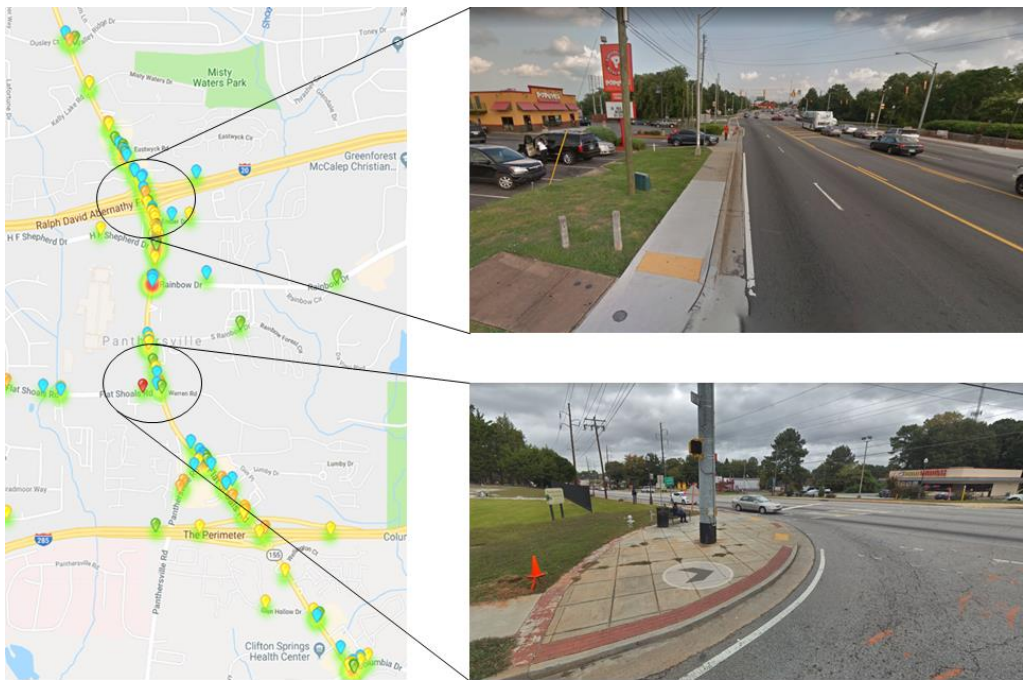
Three sites, containing six potential locations for deploying the trailer system and field testing the new algorithms, specifically for development of the “danger/safety” metric (task 4) were provided by Jack Anninos and David Adams of GDOT in January 2019. The sites include locations that have high concentrations of pedestrian incidents.

Site visits to each of the proposed locations for data collection with the trailer were performed. The goal of the site visits was to identify potential areas for placing the trailer system. This requires a significant shoulder or sidewalk space to allow pedestrians to still be able to move safely around the system. In some areas, there was not a suitable location found for placing the trailer, unless it is placed in a parking lot or in a turning center lane. The proposed locations and potential trailer placement are shown in the images below (Figure 1-3).



Memorial Drive and Kensington (Top), Memorial Drive and 278 (Bottom)

Figure 1 - Site 1



Candler @ I-20 (Top), Flat Shoals and Candler (Bottom)

Figure 2 - Site 2





J.E Lowry @ L.E Boone (Top), J.E. Lowry @ M.L.K (Bottom)

**Figure 3 – Site 3**

Due to timing and construction along some of the proposed routes, two of the six locations were available for data collection and testing with the trailer. The locations with collected data include the intersection of Joseph E. Lowry and L.E. Boone, and the intersection of Joseph E. Lowry and Martin Luther King shown above in Figure 3.

## **2) Improve reporting system capabilities**

The vast majority of work on this project consisted of designing and implementing a new software system capable of detecting and tracking pedestrians, cyclists, and vehicles simultaneously. Unfortunately, the commercial tracking solution used previously was unable to meet the requirements, as it had no ability to track cyclists, and performance was very slow. Further justification for moving to a new detector/tracker is provided in detail in the following sections.

### **Detector and Tracker Selection**

In the first quarter of 2019, several different deep neural networks were evaluated in detail. These included Resnet50 [7], Resnet101 [7], Yolov3 [8], and Detectron Mask R-CNN [9]. At the end of the evaluation, Yolov3 was chosen as the network that contained the best balance between performance and processing requirements. The following sections summarize a technical document generated detailing the final implementation of the detection network and the associated tracking algorithms in order to track pedestrians, cyclists, and vehicles.

### ***Yolov3 Finalization***

Yolov3 trained on COCO dataset (80 classes) was the preferred neural network selected based on the evaluation. The performance was not as good as the heaviest Detectron Mask R-CNN in terms of precision, but the run times were more favorable for equivalently sized networks. Also, the neural network part of Yolov3 was more readily optimizable for faster run times. That is, Mask R-CNN has some components which are not easily parallelizable, especially on a GPU, leading to longer processing times. Figure 4 illustrates the processing time of various networks that were tested including the original commercial (SLR) solution used in the previous system.

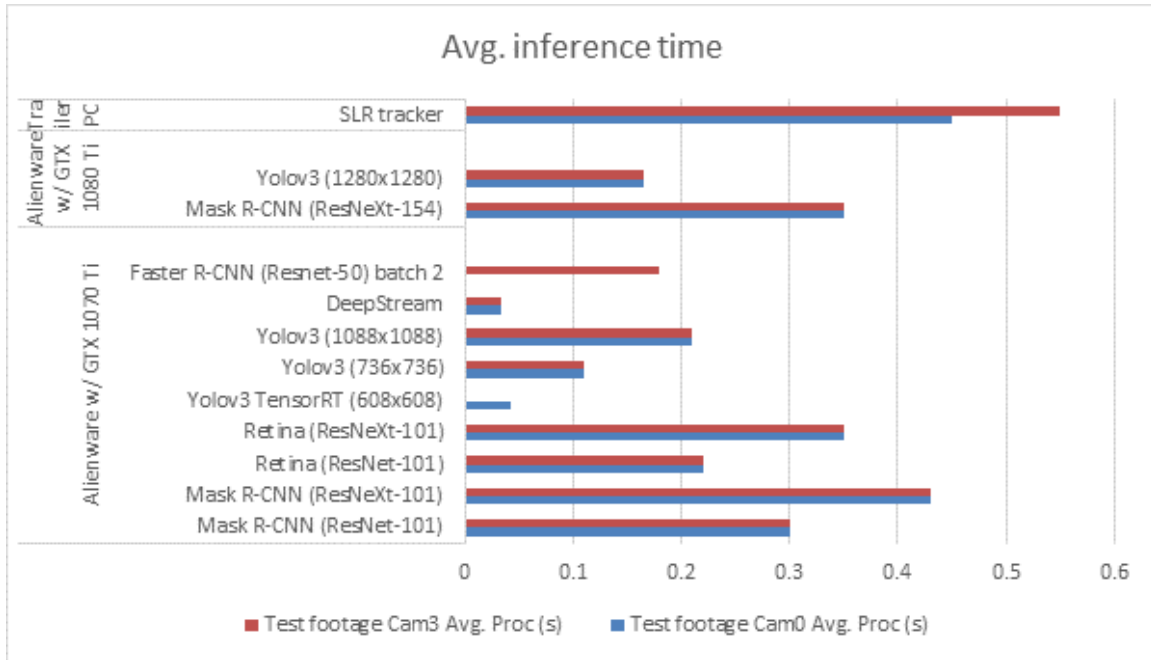


Figure 4 - Average Inference Time Comparison for Multiple Detectors

### Explore Use of TensorRT

TensorRT [10] is an SDK provided by NVIDIA that allows running of deep neural networks on NVIDIA GPUs with device-specific optimization. That means, once a deep neural network (here YOLOv3) is realized in TensorRT, it could then run on different NVIDIA GPUs and take advantage of some hardware, for instance, Tensor Cores on the newer GPUs for performing high-speed mixed precision compute, whereas GPUs that do not support this could still take advantage of other optimizations provided directly by NVIDIA. Our initial evaluations showed that YOLOv3 at a resolution of 608x608 ran almost double the speed – ~35 ms/per frame, compared to ~65 ms/per frame with the latter running on NVIDIA GPUs with CUDA and CUDNN enabled.

### Training YOLOv3

YOLOv3 training was evaluated starting with a pre-trained backbone darknet-53 (trained on imagenet), at higher resolutions than default resolutions, with COCO 2014 dataset [11]. Only image classes pertinent to pedestrians, two wheelers, and four wheelers were used. Training took approximately 4-5 days each. The performance of the newly trained network was close to

publicly available weights in terms of detecting objects, but the object confidences were not very good. Longer training could solve this problem. However, it was realized that evaluating the detector for 80 classes was not too different than evaluating for three classes since most of the last part of detection can be optimized for parallel computing on a CPU. A 1-2 millisecond difference was observed. Given this, the publicly available Yolov3 weights were used, but the realizations were changed with different network resolutions when operating in TensorRT.

### ***Tracker Evaluations***

While several tracking strategies were evaluated, a traditional tracking-by-detection paradigm is used. That is, the detector provides high-quality detections of pedestrians and vehicles, and then the tracker arranges them into coherent trajectories of subsequent bounding boxes.

The backbone of the tracker is a common Kalman filtering framework [12], preceded by an efficient linear assignment optimization to assign new detected bounding boxes to tracked bounding boxes based on Intersection-over-Union (IoU). Following that, trajectories that have sufficient confidence (Mahalanobis distance of differences of state variables) are retained. In case of pedestrians, additionally, a deep neural network [13] is used to reevaluate detected associations, in case pedestrians get blocked briefly by vehicles or a group of people are moving together having bounding boxes very close to each other.

### **Detector and Tracker Implementation**

After evaluation and selection of the detector and tracker components, the software system was designed and implemented as detailed below.

#### ***Overview of “Multimon” (multiple tracker)***

The whole pipeline is designed to be a replacement for the previous commercial tracker, but with additional capabilities. So, in that sense, it takes in similar inputs – series of images – and outputs trajectories. The outputs are then written to a database for post-processing, such as for analytics.

The module is written completely in Python, taking advantage of software and libraries such as TensorRT, numpy, scipy, and Tensorflow.

### **Detector Sub-module**

The image detector is a YOLOv3-based object detector network, completely realized within TensorRT. Currently, it is setup to process only one image at a time, but it could be easily setup to process several images at once, which takes advantage of further parallelization on a GPU. The module expects the deep neural network parameters stored as a weights file (ONNX). The weights can be exported at a required configuration and resolution from trained network parameters using a script. These ONNX weights are then used to create a TensorRT engine, which is custom-built for the specific NVIDIA GPU present on the computer, to take full advantage of all available GPU optimizations. If rerunning the network, it can use a cached version of the TensorRT engine to save time.

The detector expects an image, and produces bounding boxes, classes, and their confidences for each image. The input image is operated on by several convolutional layers, which produces an intermediate dense representation for the entire image. This dense representation is then up-sampled (with transposed convolution operations) to the input resolution, and at three of the up-sampled resolutions, the fitnesses for all classes and their bounding boxes are evaluated at each location on the up-sampled grid. In other words, the detector operates on the input image at effectively three different scales to produce detections. These are then filtered across the scales to produce only plausible detections, from each class.

### **Tracker Sub-module**

This part is largely based on deepsort [13]. Each type of tracker expects bounding boxes corresponding to the type of object it is supposed to track. A Hungarian assignment is done to assign incoming bounding boxes to tracked ones. Then, the Kalman filter updates its states

corresponding to new detections. Additionally, for the pedestrian tracker, a deep learning-based affinity metric is used to identify distinguishing features from new detections in case pedestrian detections were lost briefly due to occlusions. Each tracker is individually optimized to parallelize on the CPU to obtain better run times, especially when hundreds of objects are present in an image. The outputs of each tracker are unique tracking IDs for each new trajectory, along with their predicted bounding boxes.

### Configuration Parameters

The detector and tracker have several configuration parameters that can be changed simply by altering a configuration file. Some common configuration parameters are paths to network parameter (weights) file, input resolutions, object detector minimum confidence threshold, and number of frames to predict bounding box. A diagram illustrating the process flow of the new tracker is provided below (Figure 5).

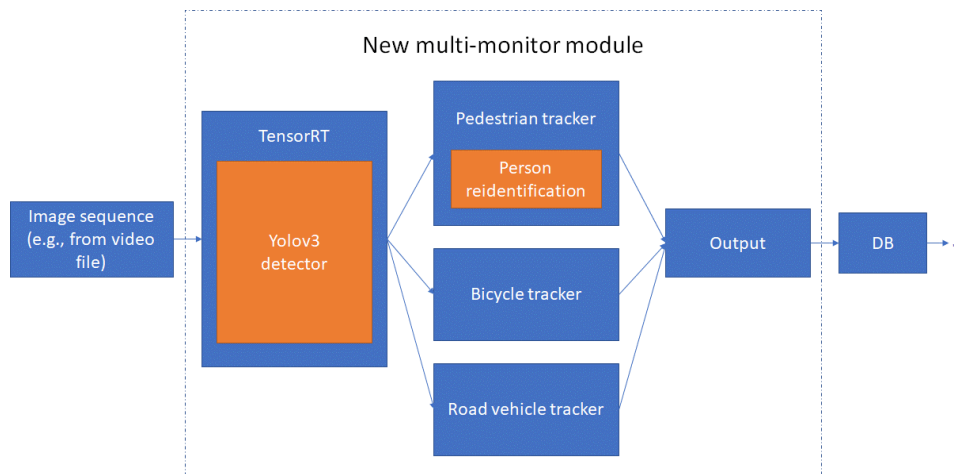


Figure 5 - Multimimon Processing Pipeline

Performance of the new detector and tracker is detailed in the following sections.

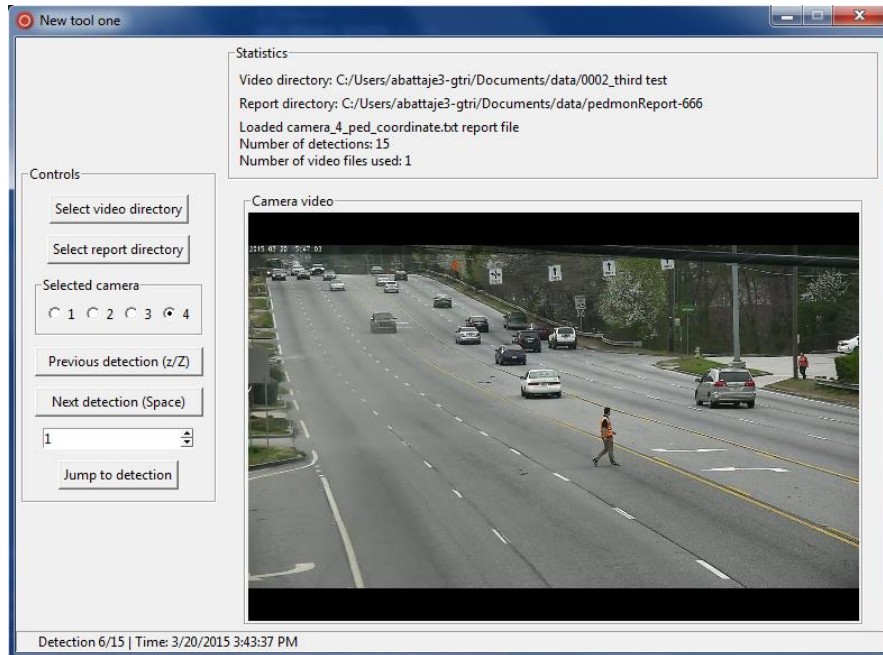
#### a) Add ability to view pedestrian crossings in video immediately using detection database

A stand-alone software tool was developed giving the ability to view pedestrian crossings in video data using the detection database. This was achieved by adding a field to the database to track the frame number in each associated video for each pedestrian trajectory. This also allows a user to quickly access still images from the video clips containing pedestrian crossings. The current database layout is shown in Figure 6 below. As described earlier, a new field was added to the Pedestrian Data table (right hand side) that includes the frame number in the video containing the crossing data.

Table Site_Info	Table Pedestrian_Data
Fields: <ul style="list-style-type: none"> <li>• Site_ID</li> <li>• Site_Desc</li> <li>• Start_Data_Time</li> <li>• End_Date_Time</li> <li>• File_Directory</li> </ul>	Fields: <ul style="list-style-type: none"> <li>• Ped_ID</li> <li>• Site_ID</li> <li>• Frame_Number</li> <li>• Trajectory_X</li> <li>• Trajectory_Y</li> <li>• Height</li> <li>• Width</li> <li>• File_Name</li> </ul>

**Figure 6 - Current Database Tables and Fields**

The new tool allows for immediate viewing of video clips containing pedestrian trajectories validated from the report generation tool. Initial attempts were done to navigate to sections of video with OpenCV computer vision libraries, but it was found that this operation proved to be very slow since OpenCV is not optimized for video viewing. Instead, libVLC (core library behind VLC media player) was employed to jump to sections of the video marked by different pedestrian crossings. This allows for rapid viewing of video clips containing pedestrian crossings as reported by the report generation tool. This tool allows better evaluation of the accuracy of the system as the user can now immediately view the video clips for characterized pedestrian crossings and identify issues with false detections (such as with cars). A screenshot of the playback tool is shown in Figure 7.



**Figure 7 - Screenshot of Video Playback Tool**

This tool allows the user to select the directory containing the video data as it is anticipated that the videos may not necessarily be on the same computer as the results generated from the report generation tool. The user then selects the directory associated with the output data from a report generated by the report generation tool (containing the report .csv file). The user can then jump through detections in sequence, or select a particular detection given the pedestrian ID if it is already known.

In addition to the video playback tool, modifications were made to the report generation tool to display an image from the video data containing the pedestrian crossing for each pedestrian. The previous video tool simply loaded the first frame from the video and overlaid pedestrian trajectories. Now the tool allows for immediate viewing of the actual pedestrian as they are crossing in addition to the trajectory overlaid on the image. This works similar to the video playback tool mentioned above, but differs in that it navigates to the part of the video section in between a pedestrian crossing and saves an image of this point in video onto a hard disk. To



accomplish this task, FFMPEG library is integrated into the report generation software. Figure 8 shows one such image as generated and stored by the software.



**Figure 8 – Sample Image Generated by Report Generation Tool Containing Trajectory and Pedestrian**

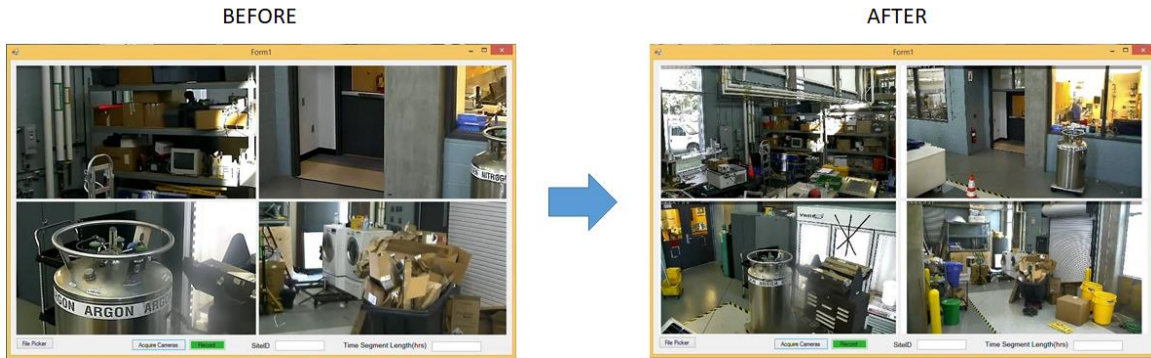
### **b) Refine the user interface based on feedback provided from GDOT**

Based on feedback provided from GDOT field operators, improvements were made to the video recording tool to ensure proper activation of the cameras on the trailer system. Specific details are described in Section 5.

The system was fully demonstrated to the new pedestrian coordinator at the GDOT in April 2018. During this time, the various tools were demonstrated and the report format and visualizations were examined. The current design was deemed acceptable from a usability perspective.

Minor improvements were made to the video recording tool on the trailer system. Primarily, this consisted of modifying the image views on the display to show the entire field of view of the cameras. Previously, the images presented by the program only showed a portion of each camera view. This allows for immediate validation of the camera field of view before recording is started and simplifies the process. A before and after shot of the user interface is shown in Figure 9.

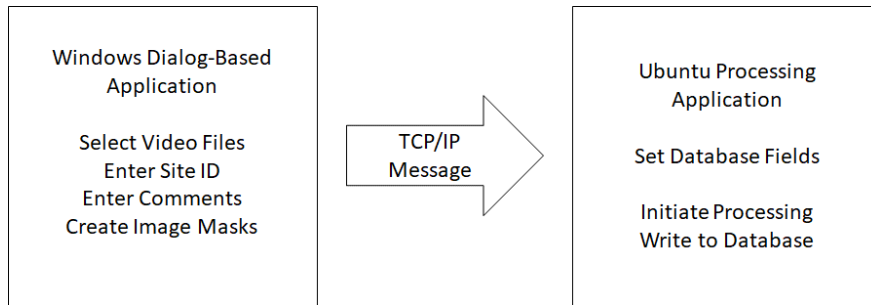
Notice the zoom level change from left-to-right.



**Figure 9 - Before and After User Interface Screenshots**

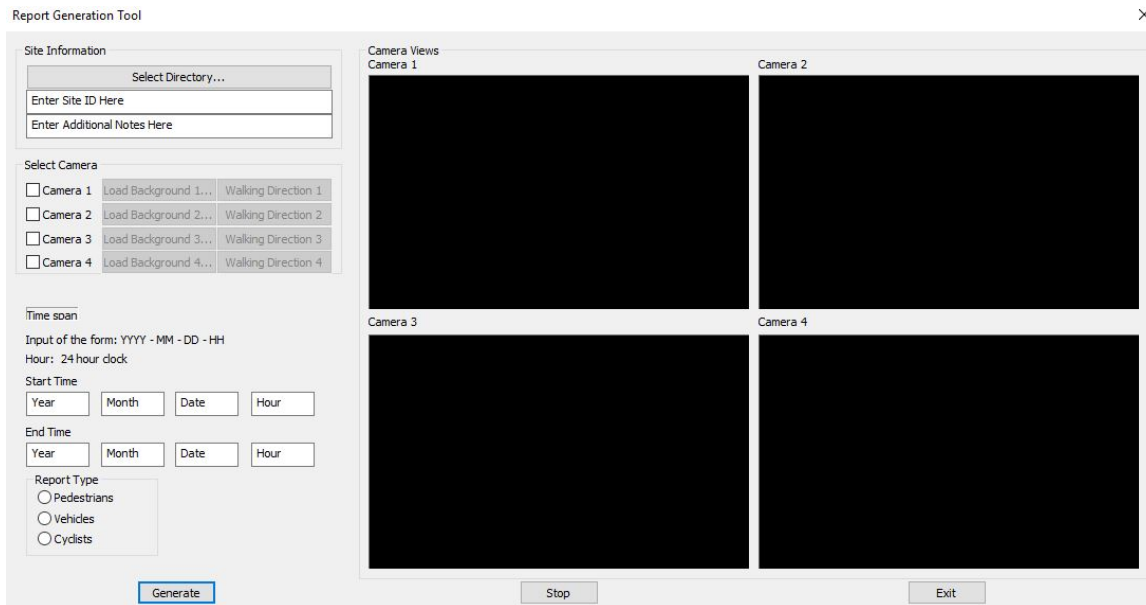
Due to the significant changes in the tracker that requires a Linux-based (Ubuntu) operating system, modifications were made to the processing front-end in the Windows-based program. The previous tracker ran in a Windows environment, so the tracker software was developed in a Windows dialog-based program. This program allowed for configuration of the locations and initiated the processing engine. For the most part, the same dialog is used and all the changes to the functionality are “hidden” to the user.

The new operation of the Windows interface allows for the same configuration operations for each site, i.e., entering site ID information, comments, and creating optional image masks. The primary change now is that the actual tracker processing is performed on a non-windows system. Therefore, the Windows program will initiate processing via passing a message to the Linux-based processor to initiate operation. The diagram below (Figure 10) illustrates the new configuration at a high level.



**Figure 10 - Windows User Interface Operation**

Final additions to the interface include more robust status indicators via a status-box, and reporting of any processing errors. In addition, a second option for processing single videos from sources other than the trailer-mounted recording system was added. A screen-shot of the front-end user interface is provided below (Figure 11). User manuals for all software tools, including this one, are included in Appendix 1.



**Figure 11 - New User Interface Screenshot**

**c) Explore the ability and accuracy for the system to identify cyclists and pedestrians separately**

Previous data captured from the trailer-mounted video system was exhaustively reviewed to extract video of cyclists for processing. This video was used as the initial input to developing approaches for identifying cyclists separate from pedestrians. The previous operation of the system does not discriminate between a cyclist and a pedestrian. SLR Engineering (commercial tracking library) and in-house custom algorithms were explored to enable classification of pedestrians and cyclists individually. A sample image of a cyclist captured by the system is shown below (Figure 12).



**Figure 12 - Sample System Capture of a Cyclist**

Additional cyclist video data was obtained on the Georgia Tech campus for testing of algorithms. Figure 13 illustrates a cyclist captured in the data collected on campus and processed with the previous commercial tracker. The blue boxes denote identified pedestrians, showing the cyclist detected (as a pedestrian) by the detector. The yellow lines are their trajectories; gray lines and dots are pedestrians that are still being processed for validation. Unfortunately, the previous tracker implementation did not support detection of cyclists. This was the primary reason for the new detection and tracking solution utilizing Yolov3.





**Figure 13 - Sample Image from Campus Data Collection Containing a Cyclist**

The new Yolov3-based tracker has the ability to independently detect cyclists. The new detector and tracking algorithms described in Section 2 above allow for this capability. An image of the tracker output from an intersection with heavy cyclist activity is shown below (Figure 14-15).



**Figure 14 - Cyclist Track Results**



To characterize accuracy of the cyclist detection and tracking algorithms, twenty-eight 30-second video clips containing cyclists were randomly selected from the pool of videos collected by the system and acquired via the GDOT camera feeds. Each of these clips contained at least one cyclist. The software was run on the video clips and reports generated. The system detected 29 of the 32 cyclists and the report generator counted 37 cyclists. The discrepancy in the total detected and the count was a result of double counting several of the cyclist trajectories.

Accurate counts for reports are something that needs to be addressed with the current system.

Table 1 summarizes the detection and tracking accuracy for cyclists.

**Table 1 - Cyclist Detection and Tracking Accuracy**

Ground Truth	Detected	Detection Accuracy	Count	Count Accuracy
32	29	91%	37	84%

**d) Explore the ability to determine GPS coordinates of crossings to allow data to be plotted in a GIS system**

An approach for extracting GPS coordinates from 2D image data was implemented and tested. This approach requires placement of a large calibration target in the video to allow for calculation of the extrinsic parameters of the camera system. This can allow for coarse estimation of the 3D position of objects from a 2D image.

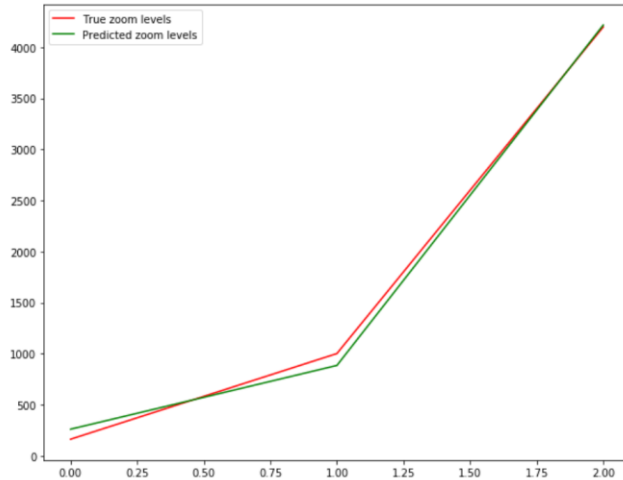
Rigorous testing and algorithm development for determining the 3D position of pedestrians with respect to the 2D camera location was carried out in Quarter 4, 2018. This required calculating intrinsic and extrinsic parameters for the camera systems. Intrinsic parameters describe lens characteristics such as focal length, sensor format, and principle point. All parameters characterize how an image is captured for that particular camera. Extrinsic parameters describe

how the 3-dimensional world is related to the 3D camera coordinates as projected onto a 2D frame. This allows a mapping between 2D image coordinates, and real-world 3-dimensional position. For a detailed summary and analysis on the technique, see Appendix 2.

Calibrating a single camera and lens is a laborious process. It requires using a hand-held target with a pattern (usually a checker-board) to be captured at several locations in an image; sometimes several hundred images are required per camera/lens combination. Software algorithms then take these images and calculate the intrinsic parameters automatically. This process is complicated by the fact that the trailer system camera location and zoom can be physically changed. For a perfect system, the lens intrinsic properties must be calibrated at every discrete zoom level, which is unrealistic to achieve due to the amount of labor required.

The team explored the relationship between the camera intrinsic parameters and zoom levels to establish the possibility of defining a function to estimate camera intrinsic properties at different zoom levels. This was achieved by manually measuring intrinsic camera values at three discrete zoom levels. Using these three points, the team was able to identify a linear relationship between zoom level and intrinsic parameters. Using this, they built a model to predict intrinsic parameters at different zoom levels. Results from testing the camera model are shown in Figure 17.

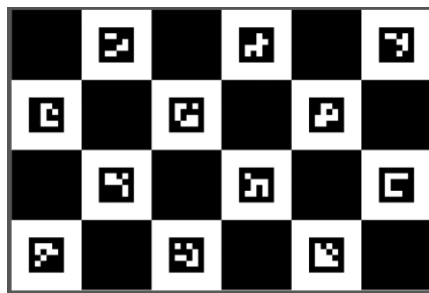




**Figure 17 - Predicted Zoom vs. Actual Using Camera Model**

As can be seen in the figure above, the camera model is able to accurately predict the focal length (and associated intrinsic properties) given an arbitrary zoom level as reported from the camera. A technical document generated as part of this testing is included in Appendix 3.

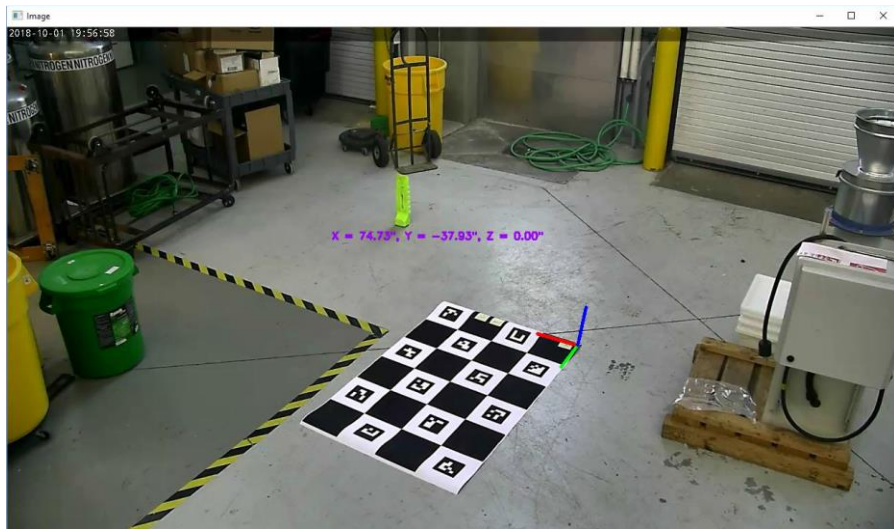
Extrinsic camera properties are solved similar to intrinsic properties, where a calibration target is utilized in a single image. Another algorithm identifies the target, and given the intrinsic parameters, is able to solve the extrinsic parameters required for characterization of 3D position from the 2D image. A picture of the calibration target is shown in Figure 18.



**Figure 18 - Calibration Target for Camera Extrinsic Parameters**

Given camera extrinsic properties, a plane can be projected into the image frame along the axis of the calibration target. This plane describes the (non-occluded) ground location in 3D space as projected into the 2D image frame. Herein lies one of the fundamental limitations of the current

algorithm, it assumes that the road surface on all sides of the camera system is relatively flat and not undulating (such as on hills). Current work is focused on solving the easier problem, that of flat roads, before augmenting the algorithm with more dynamic topologies. A sample image of the calibration target as taken in the lab is shown in Figure 19. The plane defined by the calibration is illustrated in the three axes along the calibration target's upper right corner. The red and green arrow define the x and y axis of the plane, with the blue arrow being normal to the plane.



**Figure 19 - Camera Extrinsic Target**

Several tests were carried out to validate the 3D position estimation algorithm. Technical documents for these tests are available on request. For this testing, the extrinsic camera properties were used to estimate the location of a target in subsequent image frames. Table 2 and Table 3 highlight results from two of these tests.

**Table 2 - Short Range Position Estimation Results**

	Ground truth	Without distortion			Accounting distortion		
		Estimated	Error	Relative error percentage	Estimated	Error	Relative error percentage
<b>1</b>	[59, -35, 0]	[63.68, -34.01, 0.0]	[4.68, 0.99, 0.0]	6.97	[63.33, -33.42, 0.0]	[4.33, 1.58, 0.0]	6.71
<b>2</b>	[-3, 16, 0]	[-3.61, 16.62, 0.0]	[-0.61, 0.62, 0.0]	5.34	[-3.53, 16.1, 0.0]	[-0.53, 0.10, 0.0]	3.31
<b>3</b>	[-61, -48, 0]	[-59.7, -43.33, 0.0]	[1.30, 4.67, 0.0]	6.24	[-59.43, -43.74, 0.0]	[1.57, 4.26, 0.0]	5.84
<b>4</b>	[72, -36, 0]	[74.73, -37.93, 0.0]	[2.73, -1.93, 0.0]	4.15	[74.21, -36.09, 0.0]	[2.21, -0.09, 0.0]	2.74
<b>5</b>	[-15, 6, 0]	[-15.8, 5.1, 0.0]	[-0.80, -0.90, 0.0]	7.45	[-14.88, 6.17, 0.0]	[0.12, 0.17, 0.0]	1.28

**Table 3 - Long Range Position Estimation Results**

	Ground truth	Estimated	Error	Relative error percentage
<b>1</b>	[132, 66, 0]	[131.11, 63.22, 0.0]	[-0.89, -2.78, 0.0]	1.97
<b>2</b>	[451.5, 66.0, 0.0]	[438.7, 58.18, 0.0]	[-12.80, -7.82, 0.0]	3.28
<b>3</b>	[535.5, 66.0, 0.0]	[516.58, 57.67, 0.0]	[-18.92, -8.33, 0.0]	3.83
<b>4</b>	[571.5, 6.0, 0.0]	[543.18, 1.93, 0.0]	[-28.32, -4.07, 0.0]	5.00
<b>5</b>	[986.5, 24.0, 0.0]	[898.77, 15.57, 0.0]	[-87.73, -8.43, 0.0]	8.93
<b>5 (4x zoom)</b>	[986.5, 24.0, 0.0]	[1124.78, 44.69, 0.0]	[138.28, 20.69, 0.0]	14.17

Data was collected using the trailer system on the Georgia Tech campus with the goal being to test the 3D position estimation algorithms. Unfortunately, the algorithm was not able to detect the calibration target in the images as it was too saturated and the image quality was too compromised. An image of the calibration target captured during data collection on campus is shown in Figure 20. The calibration target is located in the bottom middle of the image. The oblique angle in combination with the camera resolution and image compression cause the target to get identified with large error. Current work is focused on solving this issue with further real-world testing of the system and various calibration targets.



**Figure 20 - Campus Testing of GPS Location Capability**

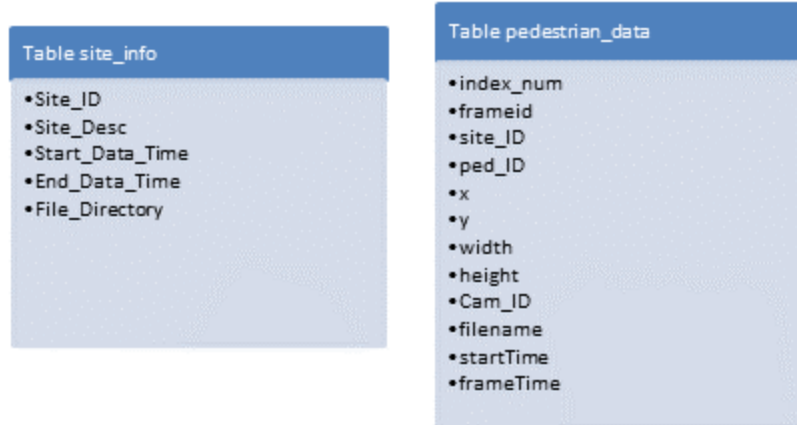
As an exploratory task, this activity has been completed for the scope of the current project. Several other research projects and open-source algorithms have shown the capability to map a flat road-surface into a GIS system in order to determine object speeds and positions in GPS coordinates. This functionality can be used by the current system in intersections and locations where the road surface is flat. However, in the case of midblocks and long stretches of road where there are hills and undulating geometry, these algorithms break down. A method for mapping uneven terrain and complicated geometry is required in order to determine global position data for objects in complex scenes. Promising approaches are currently being researched by several institutions including Microsoft [14].

#### **e) Make further improvements to system processing time**

Effort was spent examining the current database table and field layout in terms of optimization. An approach for redesign of the internal database structure was identified with the explicit goal of improving report generation time with large sets of trajectory data. For example, a single day of pedestrian crossing data can contain over a hundred thousand records (as seen from field testing the system in the past). Generating reports on these large sets of data is computationally intensive and leads to sometimes several minute waits for report generation. Standard optimization

methods were used to improve this processing time to ensure that reports can be readily generated in a reasonable amount of time.

Report generation used to be slow due to inefficiencies in the database structure. Therefore, the database was modified to include unique keys to help with fast indexing during the “select” queries from the report generation tool. Figure 21 shows the modified schema for the database. The field filenames Cam\_ID, startTime, and frameTime are used to navigate to sections of the video in the playback tool. A new field index\_num was added as a unique key that auto-increments for each new entry added to the table. The unique key allows faster access.



**Figure 21 - New Database Structure**

The improvement in performance was tested with continuous queries of different types to the database. Figure 22 shows the return times for this test. As shown, all queries for report generation, even those containing over 500,000 rows of data, completed in less than one second. This is a vast improvement from the several seconds (sometimes more than 30 seconds) of query times previously.

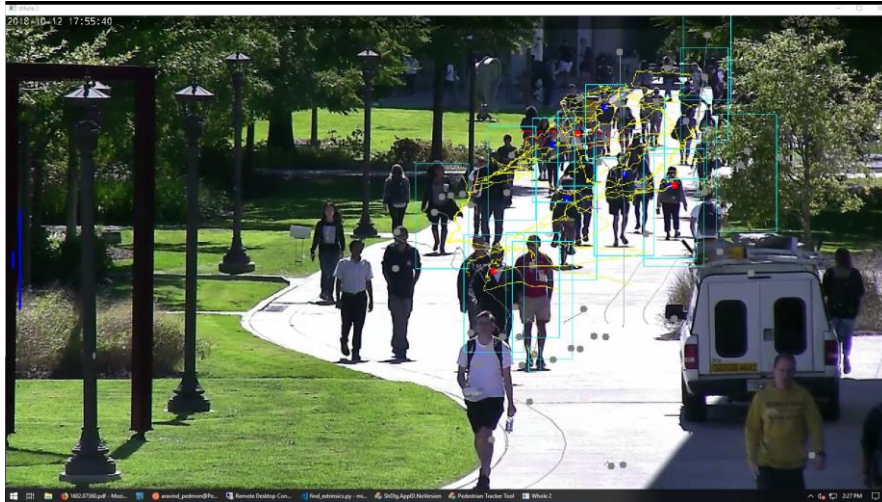
	Time	Action	Message	Duration / Fetch
✓	1 12:11:11	select * from pedmon.detection wher...	537800 row(s) returned	0.266 sec / 0.672 sec
✓	2 12:14:15	select * from pedmon.site_info	21 row(s) returned	0.000 sec / 0.000 sec
✓	3 12:28:02	select count(*) from pedmon.site_info	1 row(s) returned	0.000 sec / 0.000 sec
✓	4 12:28:02	select count(*) from pedmon.detection	1 row(s) returned	0.328 sec / 0.000 sec
✓	5 12:28:03	select * from pedmon.detection wher...	537800 row(s) returned	0.125 sec / 0.625 sec
✓	6 12:28:08	select * from pedmon.detection wher...	0 row(s) returned	0.313 sec / 0.000 sec
✓	7 12:28:08	select * from pedmon.detection wher...	0 row(s) returned	0.328 sec / 0.000 sec
✓	8 12:28:09	select count(*) from pedmon.detectio...	1 row(s) returned	0.125 sec / 0.000 sec
✓	9 12:28:09	select count(*) from pedmon.detectio...	1 row(s) returned	0.125 sec / 0.000 sec
✓	10 12:28:09	select count(*) from pedmon.detectio...	1 row(s) returned	0.125 sec / 0.000 sec
✓	11 12:28:09	select count(*) from pedmon.detectio...	1 row(s) returned	0.125 sec / 0.000 sec
✓	12 12:28:10	select count(*) from pedmon.detectio...	1 row(s) returned	0.000 sec / 0.000 sec
✓	13 12:28:10	select * from pedmon.detection wher...	0 row(s) returned	0.343 sec / 0.000 sec
✓	14 12:28:10	select * from pedmon.detection wher...	0 row(s) returned	0.328 sec / 0.000 sec
✓	15 12:28:10	select * from pedmon.detection wher...	100 row(s) returned	0.110 sec / 0.000 sec
✓	16 12:28:11	SELECT frameid', 'index_num', 'Ped_...	14570 row(s) returned	0.438 sec / 0.000 sec
✓	17 12:28:11	SELECT frameid', 'index_num', 'Ped_...	8573 row(s) returned	0.391 sec / 0.000 sec
✓	18 12:28:12	SELECT frameid', 'index_num', 'Ped_...	23816 row(s) returned	0.406 sec / 0.016 sec
✓	19 12:28:12	select * from pedmon.detection wher...	537800 row(s) returned	0.109 sec / 0.640 sec

Figure 22 - Time-Test Results from Database Queries

With the report generation tool optimized, efforts were spent examining ways to improve the previous tracking tool processing time. The tracking tool was benchmarked on the previous field test data sets and efforts were spent adjusting the tracker parameters in an attempt to optimize the processing time while maintaining system accuracy. It was found that tracking performance is highly scene dependent. For example, scenes with more complicated geometry (such as buildings) led to longer processing times per frame. In addition, the more moving objects, such as cars and pedestrians, in a scene also impacted the processing time for each frame.

New data collected at sites around the Georgia Tech campus identified additional issues with the previous commercial tracker being used for pedestrian detection. In addition to the tracker performance being negatively impacted leading to longer processing times when there are large numbers of pedestrians, the tracker accuracy falters as can be seen in Figure 23. This was found to be unacceptable for tracking at all types of varying locations and was a major factor in moving to a new detection and tracking solution.

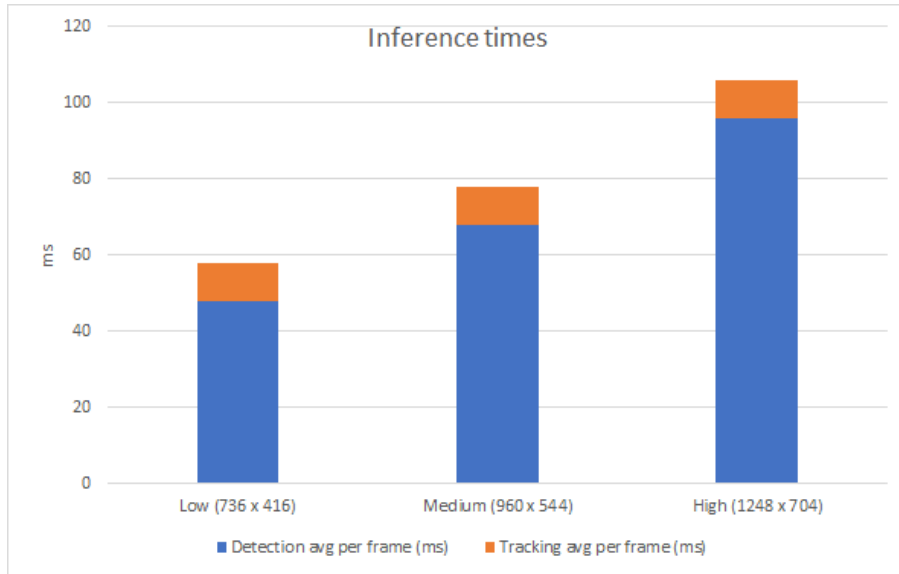




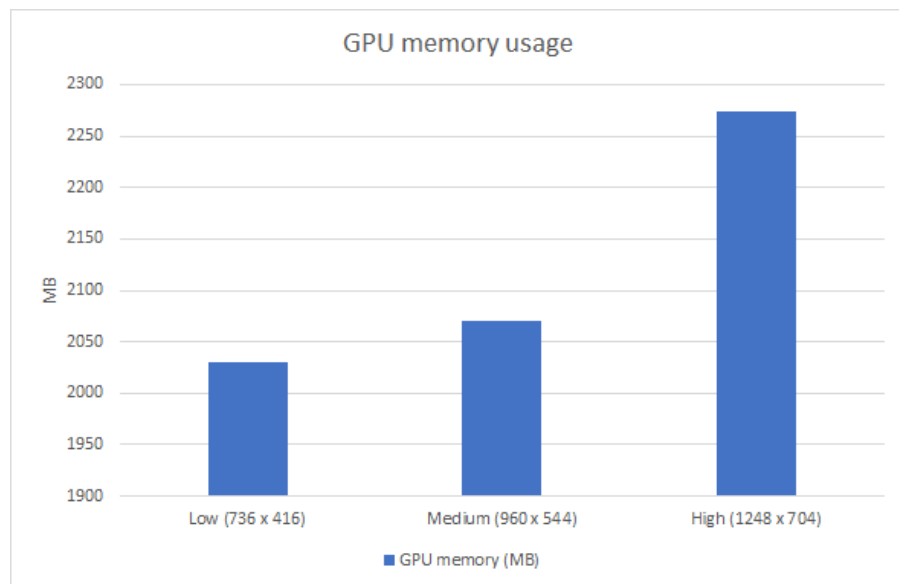
**Figure 23 - Pedestrian Tracker Accuracy Negatively Affected by High Volumes**

As part of the development of the new tracker, a new PC was purchased with modernized hardware including an Nvidia GPU. Utilizing the new GPU, the selected DNN (Yolov3) was optimized and a TensorRT implementation of the classification network was generated. This optimization allows for a lower memory footprint and faster processing times for the detection and tracking algorithms.

The detector network is created in TensorRT at three different resolutions for flexibility: low, medium, and high. The low resolution allows faster processing, perhaps, real-time, while it might miss detections that are too small, e.g., pedestrians at a distance. The high resolution, running very close to captured (720p HD) resolution, allows the capture of many more detections, but is not efficient in processing time and needs marginally more RAM on the GPU. The medium resolution network provides a nice tradeoff between the two. Figure 24 and Figure 25 summarize the processing time per frame and the GPU memory required for detection and tracking, respectively.



**Figure 24 - Inference Time for Detection and Tracking**



**Figure 25 - GPU Memory Usage**

Further modifications were made to the processing system to improve processing time. It was discovered that the current design of the software communication to the database for logging trajectory data was non-optimal due to the reuse of a single connection to the database. This means that for each trajectory written to a database, the connection must be opened, the data written, and then the connection closed. Maintaining an open connection allows for a speedup of



approximately 13 times. To further improve performance, a connection pool was implemented, allowing for multiple simultaneous connections to the database. The result of this is faster tracker performance.

Table 4 shows the times from three different tests, each with three different configurations. All the tests use a 30-second video clip, with the detector running on all types (pedestrians, vehicles, and cyclists). All the objects are detected, tracked, and written to the database. All the database queries are handled efficiently with a pool of connections to maximize query executing performance. As mentioned above, the pool of connections is maintained to avoid reconnections to the server for every write.

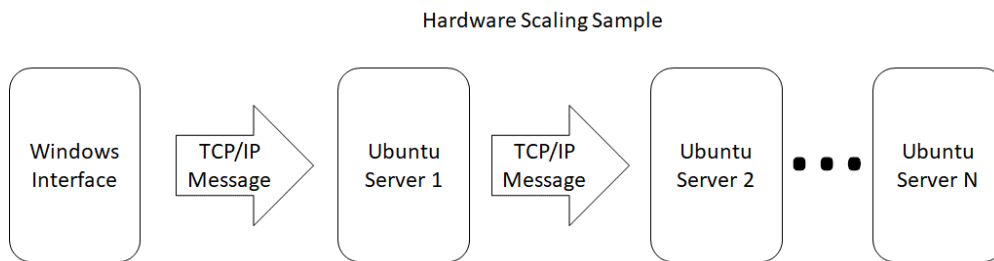
**Table 4 - Tracker Database Query Optimization Results**

Frames Processed	Total Time	Average Per Frame	Database Query Overhead	Average Overhead Per Frame
847	27.5 seconds	0.065 s	1.01 s	0.002
847	44.5 seconds	0.105 s	19.9 s	0.047
847	26.7 seconds	0.063 s	0.02 s	0.000

The three runs differ in their database connection configuration. The first two are run on a local server on Linux. The first one uses additional background processing for database handling so that detection/tracking and writing queries are handled in parallel. This only has an overhead of ~1 second, while the second test, which executes database handling serially with detection/tracking, has a total overhead of ~19 seconds. Note, the second test is analogous to the previous implementations, but is still faster because it reuses connections via pools. Without the connection pools, the overhead is ~35 seconds (not shown here). The third test uses a Windows MySQL (remote) server, along with all forms of parallelism explained above. It boasts the least amount of overhead. Therefore, the final database configuration will run the database on a Microsoft Windows system, while the tracker runs on a Linux operating system.

Currently, the system is capable of processing 15FPS video data at faster than real-time, with up to four simultaneous video streams. With the current hardware configuration utilizing an Nvidia RTX Titan video card with 24GB of memory, the system is able to process more than a dozen simultaneous video feeds. However, with higher framerate video or more than four video feeds, the system is unable to maintain real-time processing on the single PC.

If real-time processing is required for more than four videos, or for videos at significantly higher frame rates, the software is designed to allow for addition of more hardware to allow for more simultaneous processing. In this way, a system can be scaled up to allow for real-time processing of as many simultaneous video streams as necessary. Figure 26 illustrates the hardware scaling capabilities of the system.



**Figure 26 - Hardware Scaling for Simultaneous Video Processing**

### **f) Improve and characterize system ability to track pedestrians in intersections**

The system was initially deployed at locations around the Georgia Tech campus to capture pedestrians at intersections. This data was used for exploring modifications to the previous tracking system to improve the ability to accurately detect and track pedestrians in intersections.

A sample image from an intersection on the Georgia Tech campus is shown in Figure 27.



**Figure 27 - Video Data Capture at Intersection on the Georgia Tech Campus**

A new tool was created to assist in the development of routines to improve the accuracy of pedestrian detection in intersections. Previously, the system was able to achieve high accuracy specifically in midblock locations because vehicles are moving perpendicular to the direction of pedestrian crossings. This allows the system to easily remove false positives due to vehicles. Unfortunately, in intersections, both vehicles and pedestrians can be crossing in the same directions parallel to each other. Even though the system has a low false positive detection rate, the fact that there are hundreds more cars than pedestrians in a large data set means that false detections of vehicles can heavily outweigh positive detections of pedestrians.

The tool allows for extraction of detections for immediate review. These detection results include pedestrians (positive detections) and false detections on items such as vehicle wheels and fenders. Sample images of correct and incorrect classification results are shown in Figure 28.



**Figure 28 – Sample Images of Positive (Top) and False (Bottom) Detections**

As can be seen in the sample images above, there are several false detections caused by vehicles using the previous detector. The results above further illustrate the need to move to a new detector and tracker suite. The ability to simultaneously characterize vehicles, cyclists, and pedestrians using the new YOLOv3-based detector eliminates the largest source of error in the previous system, which is false detections of pedestrians due to vehicles. This capability allows the trackers to perform with high detection accuracy in various scenarios, in midblocks, intersections, and pedestrian corridors. Figure 29 illustrates output from the tracker operating in a pedestrian corridor intersection with vehicles. This image contains the actual pedestrian trajectories through a heavily used intersection.



**Figure 29 – Intersection and Sidewalk with Pedestrian Tracks**

For validation, 30-second video clips were randomly selected from the pool of all videos captured by the trailer system and obtained via other sources such as GDOT cameras and online videos. A total of 112 video clips were used for validation, of which 55 contained pedestrians in midblock crossings, and the remaining 57 contained pedestrians crossing in intersections. From both of these locations, a total of 469 pedestrians were manually identified in the videos. Of these, 450 were detected, yielding an extremely high detection accuracy of 96%. However, count accuracy is marginally worse. This is due to occlusion of pedestrians as they cross in groups, leading to excessive multiple counts. Table 5 details results from the accuracy testing for pedestrians both in and outside intersections.

**Table 5 – Pedestrian Accuracy Results**

Location	Ground Truth	Detected	Detection Accuracy	Count	Count Accuracy
Intersection	291	279	96%	334	85%
Midblock	178	171	96%	212	81%
Both	469	450	96%	546	84%



### g) Add and evaluate vehicle tracking abilities

Similar to the above sections, the new Yolov3-based detection system includes vehicle detection capabilities. A sample output from the vehicle tracker is shown in Figure 30. One can clearly see in Figure 31 the concentrations of vehicles in the individual lanes from the output of the system.



Figure 30 - Vehicle Trajectories

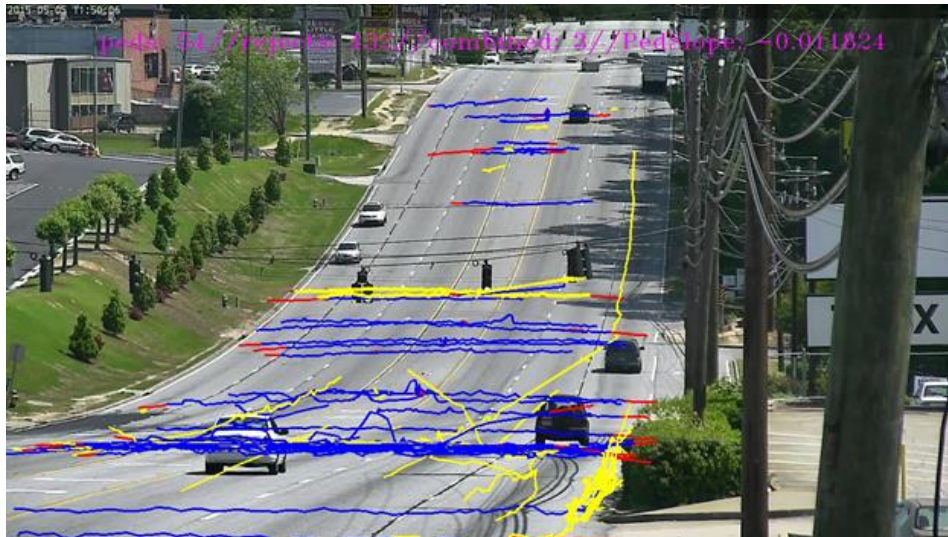
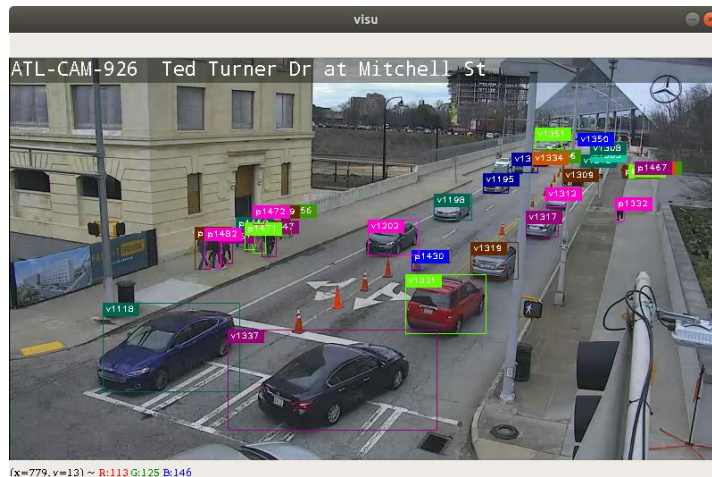


Figure 31 - Pedestrian Activity at Same Location as Vehicles in Above Image

With the completion of the new tracker implementation, significant effort was spent validating the system accuracy. In the past, when only pedestrians were considered, this was a substantially

easier task. Now that the system includes vehicles and cyclists, validation becomes a much harder prospect, particularly as it relates to manual assessment. The vast number of vehicles make manual assessment very difficult and error prone. Figure 32 shows a screenshot of the tracker output that illustrates performance on a common scene.



**Figure 32 - Vehicle Tracker Sample Screenshot**

To characterize accuracy of the vehicle detection and tracking algorithms, fourteen 30-second video clips containing vehicles were randomly selected from the pool of videos collected by the system and acquired via the GDOT camera feeds. Each of these clips contained several vehicles. The software was run on the video clips and reports generated. Manual assessment counted a total of 163 vehicles. The detector successfully identified 158 of the vehicles, of which 155 were counted via the report generation tool. The high accuracy of the detector and tracking is due to the relatively easy nature of detecting vehicles as opposed to pedestrians and cyclists which have significantly more variety in features. Table 6 summarizes the results from the accuracy assessment.

**Table 6 - Vehicle Detection and Tracking Accuracy**

Ground Truth	Detected	Detection Accuracy	Count	Count Accuracy
163	158	97%	155	95%

### **3) Explore data processing with different video sources**

The purpose of this task was to add the capability for the detection and tracking software to work with video sources other than those captured on the trailer recording system. Previously, the software required very specific video formats and naming conventions in order to process and generate reports. Modifications were made to the user interface and the processing engine to allow for processing of videos from any source.

Working closely with Dr. Randall Guensler and Dr. Angshuman Guin from the School of Civil and Environmental Engineering at Georgia Tech, video was obtained of several locations containing different intersections and streets throughout Atlanta. Some of these sites are already shown in the images above. Two early results from the new tracking system are shown below. In Figure 33, the result from processing a heavy pedestrian corridor is shown. This data represents tracks of pedestrians leaving an event at the Mercedes Benz Stadium. While several tracks accurately show some pedestrians walking through the traffic lanes, most of the pedestrian tracks in the road were the result of the detector identifying occupants inside vehicles as pedestrians. In Figure 34, showing vehicle tracks in the same location, some errors in the vehicle tracker resulted in several straight-line trajectories off to the sides of the roadways. Both errors were addressed as described in the following paragraphs.





Figure 33 - Intersection with Heavy Pedestrian Activity

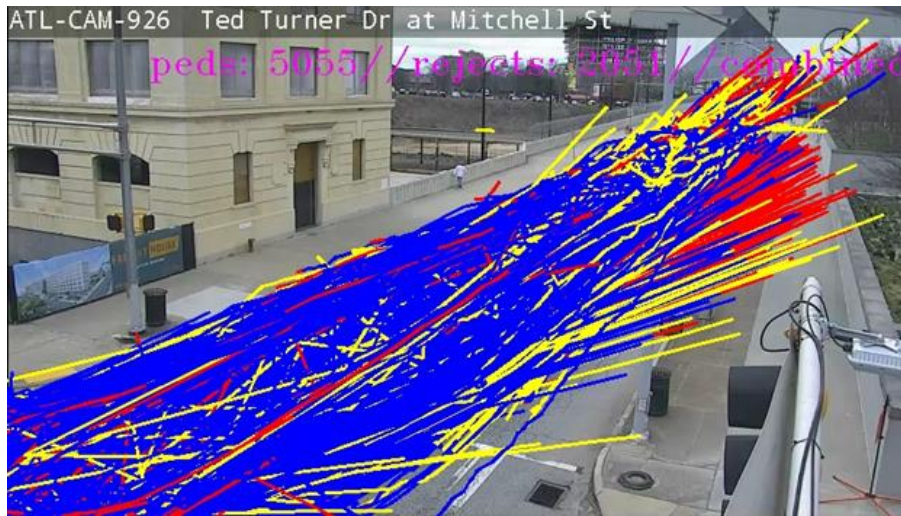
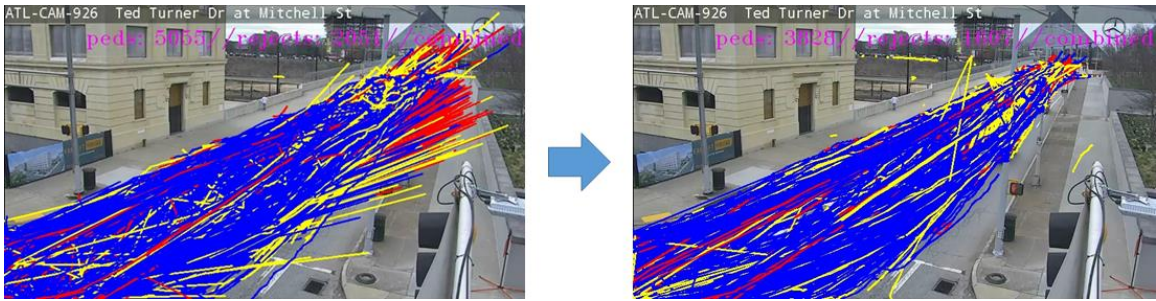


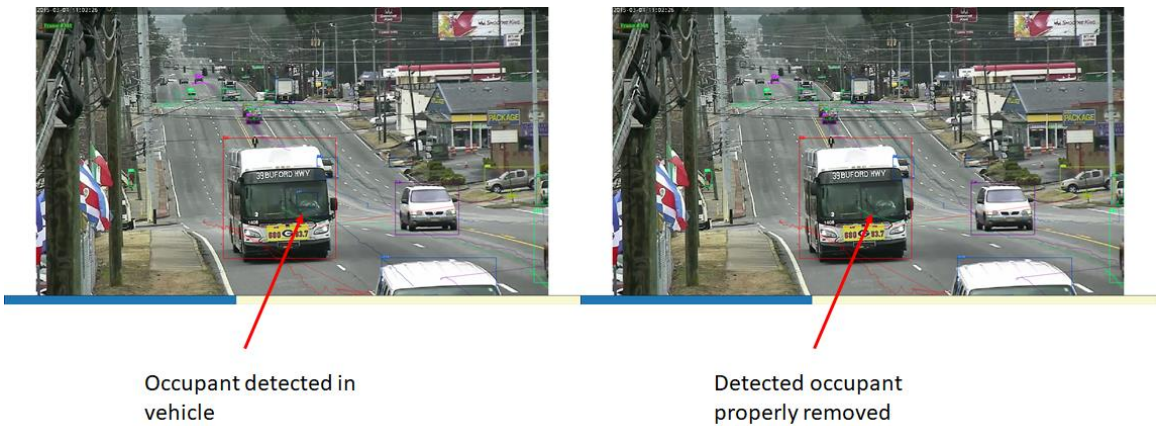
Figure 34 - Vehicle Tracks Illustrating Tracker Errors Due to Configuration

The errors with the vehicle tracker shown above were related to non-optimal configurations for the new tracker. Upon completion of the new tracking software, significant effort was spent optimizing the tracker filter settings to maximize performance. A sample of this improvement is presented in Figure 35. In the screenshots, you can plainly see errors in tracking performance in the left hand image that have been eliminated in the right hand image. In this report, several hundred vehicles were detected and tracked. This also illustrates some of the challenges when it comes to validation of system accuracy.



**Figure 35- Fixed Tracker Errors Due to Configuration Settings**

A second issue with the new tracker was one of almost “too good” performance. The detector is powerful enough to identify occupants inside vehicles. This means that the trackers are able to identify people inside cars, but unable to separate them from people outside cars. To solve this, initial logic was added to the trackers to remove detections that are located inside detected vehicles. Figure 36 shows a sample of a person detected inside a bus, and then the same person not detected after adding the new logic to the tracker.



**Figure 36 - Vehicle Occupants Counted as Pedestrians Before and After**

Several sample screenshots from the new detector output taken from various video feeds are included below (Figure 37-40):







Figure 39 - Sample Detector Output

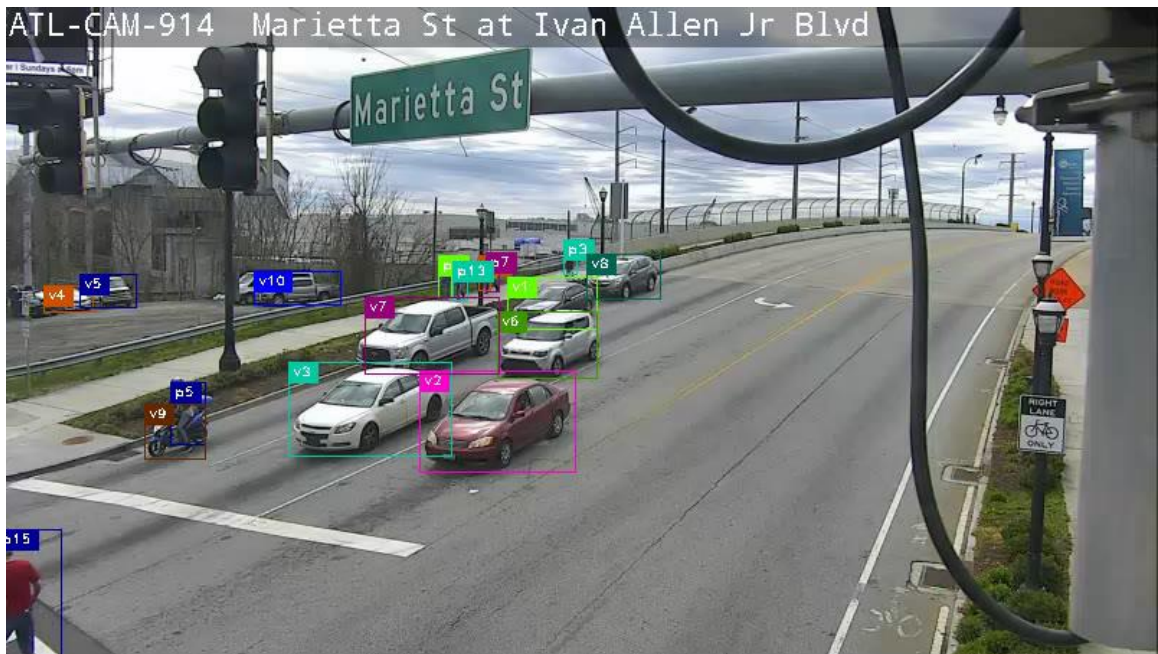


Figure 40 - Sample Detector Output



#### 4) Evaluate the ability to automatically perform incident detection or generate a “danger/safety” metric based on vehicle and pedestrian interactions

Significant effort was also expended developing a “danger/safety” metric. A comprehensive literature survey was conducted, which can be found with summaries attached in Appendix 4. A software tool able to query reports generated by the report generation tool allows for automatic extraction of various vehicle/pedestrian/cyclist interactions. A software manual for this tool is located in Appendix 1.

A detailed description of the method for extracting potential interactions between pedestrians and vehicles is described in Appendix 5. The tracker output can be used to analyze pedestrian and vehicle conflicts. The following section details a sample conflict analysis for a processed video file. Table 7 and Figure 41 below depict the outputs of the analysis.

Table 7 - Conflict Detection Sample Data

	PET	X	Y	Ped ID	Ped Int Time	Veh ID	Veh Int Time	Category
0	-3.666667	122.500000	500	12	73.233333	255	69.566667	Horizontal
1	3.470073	206.040146	497.307	12	71.610219	375	75.080292	Horizontal
2	11.533333	526.500000	942	51	120.633333	574	132.166667	Vertical
3	5.541667	391.000000	363	78	145.391667	647	150.933333	Horizontal
4	3.533333	612.500000	363	78	147.600000	649	151.133333	Horizontal
6	5.166667	827.000000	607	81	161.133333	576	166.300000	Horizontal
5	6.300000	683.000000	362	86	148.100000	649	154.400000	Horizontal
7	-1.911111	582.250000	862	177	275.161111	1327	273.250000	Horizontal

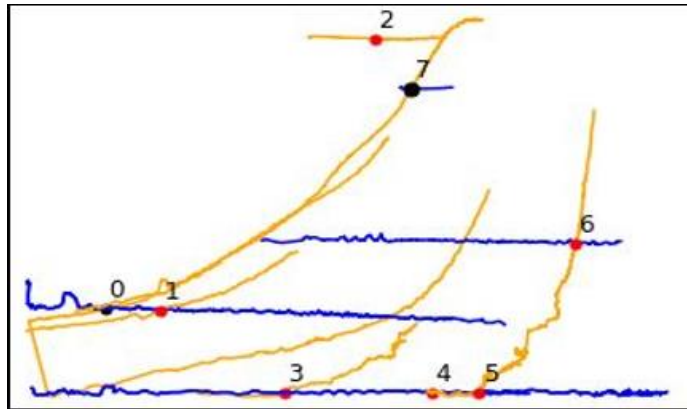


Figure 41 - Conflict Plot

The conflict table shows the post-encroachment time (PET) of the conflict, the x and y coordinates of the conflict location, the unique pedestrian ID, the pedestrian intersection time, the unique vehicle ID, the vehicle intersection time, and the conflict category.

The PET represents the time difference between pedestrian and vehicles at an intersecting point. A negative PET means that a pedestrian passes behind a car. A positive PET means a pedestrian passes in front of a car, which is a more severe type of conflict. Black dots on the plot depict negative PET, and red dots depict a positive PET. A conflict occurs when a unique pedestrian's trajectory intersects with a unique vehicle's trajectory within a certain time window. A time window is applied to filter out less meaningful interactions like a vehicle passing over a pedestrian's trajectory after 30 seconds.

The intersection coordinates for a given conflict is depicted in the x and y columns of the conflict table (Table 7). The pedestrian intersection time ('Ped Int Time' column in the table) represents the time a pedestrian intersects the conflict points, and vehicle intersection time ('Veh Int Time' column in the table) represents the same for vehicles. These values are calculated by interpolating between the conflict point and individual pedestrian and vehicle trajectories. The times are reported in video time, so the values represent when a pedestrian or vehicle are at a conflict point.

Conflict category represents whether the pedestrian was travelling horizontally or vertically.

Figure 42 shows a pedestrian with ID 12 crossing in front of vehicle 375 and behind vehicle 255.

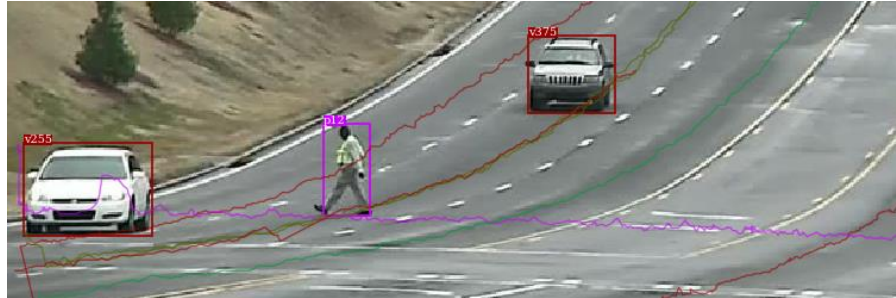


Figure 42 - Conflict Validation

Table 8 shows a negative PET between pedestrian 12 and vehicle 255, and a positive PET between pedestrian 12 and vehicle 375. Manual verification confirmed that the output PETs were accurate. Figure 43 contains the associated conflict plot.

Table 8 - Conflict Table

	PET	X	Y	Ped ID	Ped Int Time	Veh ID	Veh Int Time
0	-3.666667	122.500000	500	12	73.233333	255	69.566667
1	3.470073	206.040146	497.307	12	71.610219	375	75.080292

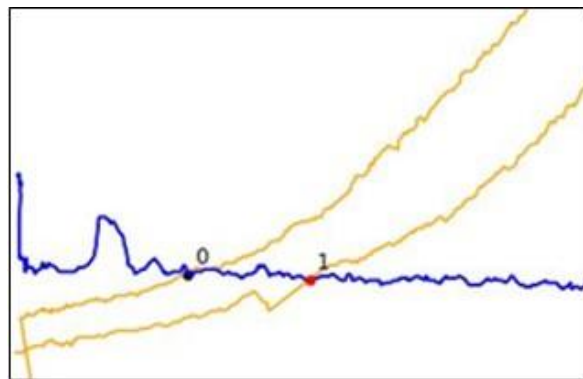


Figure 43 - Conflict Plot

In certain cases, false-positives can occur. Figure 44 demonstrates one case where a conflict was registered, but never actually happened.



**Figure 44 - False Conflict Sample**

The images above represent conflict 2 in Table 7. Pedestrian 51 is tracked first, then vehicle 574 “crossed” the pedestrian’s path. In reality, vehicle 574 crossed behind and never intersected pedestrian 51’s trajectory. This is a common occurrence that can cause false positives with the current approach.

Conflict analysis was carried out on four different sites, containing 13 different videos. The first site contains both near and far northbound and southbound views on a stretch of Buford Highway, as depicted in Figure 45. The second site consists of all four directions through the intersection of Joseph E. Lowery and Joseph E. Boone as shown in Figure 46. The third site contains all four directions at the intersection of Joseph E. Lowery and Martin Luther King as shown in Figure 47. Finally, the fourth site contains a single video captured at 10<sup>th</sup> Street and Myrtle as shown in Figure 48.





Figure 45 - Site 1 along Buford Highway



Figure 46 - Site 2, Intersection of J.E. Lowery and J.E. Boone





**Figure 47 - Site 3, Intersection of J.E. Lowery and M.L.K.**



**Figure 48 - Site 4 Located at 10th and Myrtle**

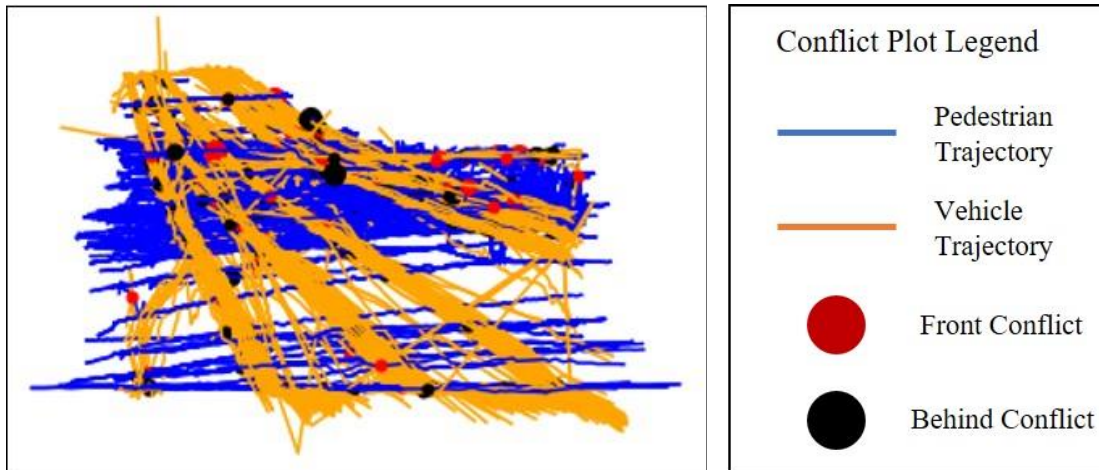
Site 1 along Buford Highway contained video data for three days. As seen from the four camera views, this area of Buford Highway does not have many crosswalks, so pedestrians must cross seven lanes of traffic with vehicles traveling at high speeds. Table 9 depicts the number of

conflicts and conflict type for each camera. In the table, behind conflicts are when the pedestrian crosses behind a vehicle, meaning the vehicle is travelling away from the pedestrian. Therefore, behind conflicts can essentially be ignored. Conflicts are when a pedestrian crosses in front of a vehicle, meaning the vehicle is travelling towards the pedestrian. Near misses are considered when the intersection time between a pedestrian and a vehicle is less than 3 seconds.

**Table 9 - Conflict Table for Site 1**

Camera	Total Conflicts	Behind Conflicts	Behind Near Miss	Conflicts	Near Miss
1	368	184	41	184	152
2	336	234	50	102	96
3	140	83	21	57	52
4	664	447	98	217	195

From the table, the camera view with the highest number of conflicts is camera 4. This data can help support an area's context; this view has a large amount of residences on one side disconnected from the MARTA stop and other amenities. The conflict plot for this camera is shown in Figure 49.



**Figure 49 - Conflict Plot for Site 1, Camera 4**

As can be seen in the plot, a significant number of pedestrians (blue lines) preferred to cross the street directly in lieu of walking to a safe crosswalk. In addition, at this particular stretch of roadway, vehicles are moving at a significant speed. While the majority of the conflicts here identified were crossings behind a car (black dots), there were still a large number of pedestrian-vehicle interactions (red dots). One poorly timed gap or inattentive pedestrian or driver would likely cause a severe incident at these speeds.

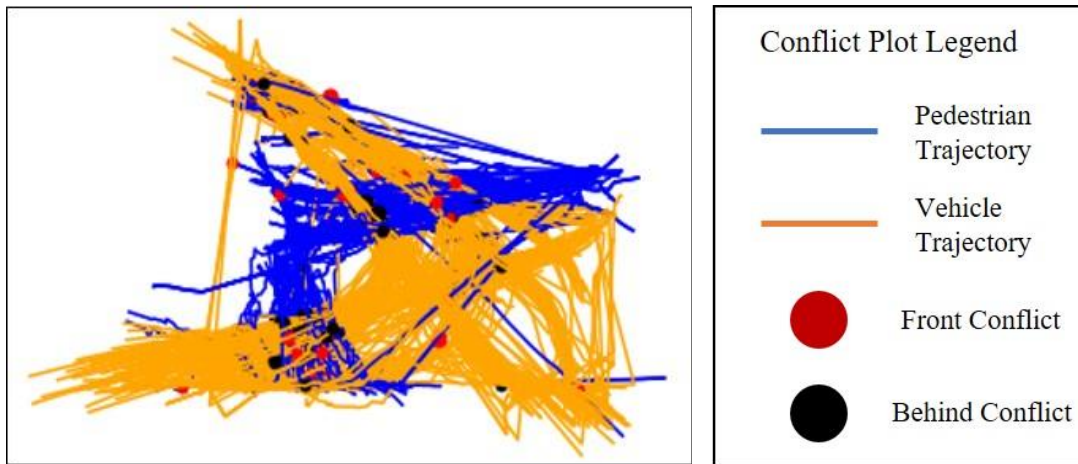
Site 2 contains five hours of video. From the different views, cameras 1 and 4 are pointed at the intersection of Joseph E. Lowery and Joseph E. Boone, while cameras 2 and 3 are directed away from the intersection. While this is a signalized intersection, conflicts between pedestrian and vehicles can still occur from permissive left and right turns, as well as pedestrians disobeying the signal head. Table 10 shows that camera 4 had the most conflicts.

**Table 10 - Conflict Table for Site 2**

Camera	Total Conflicts	Behind Conflicts	Behind Near Miss	Conflicts	Near Miss
1	256	141	31	115	108
2	277	179	35	98	93

3	14	10	4	4	3
4	321	171	31	150	136

Figure 50 shows all conflicts plotted for camera 4. Based on the figure, most conflicts occur when a pedestrian is within a crosswalk. While vehicles are unlikely to be traveling at high speeds, turning vehicles may not properly yield to pedestrians, which generates conflicts.



**Figure 50 - Conflict Plot for Site 2, Camera 4**

Camera 2 also has a significant number of conflicts (277). Figure 51 shows the conflict plot for camera 2. Camera 2’s view, like Buford Highway, has no crosswalks, yet a high number of conflicts. Examining the video, a heavy construction area causes these conflicts as construction workers cross the street to go to their personal vehicles parked in a grassy area. This information might be useful to policy makers who can encourage developers to implement pedestrian-vehicle conflict mitigation efforts. Overall, the conflict tool can highlight unexpected areas of conflict, such as a construction area, in addition to providing information about pedestrian safety at intersections.

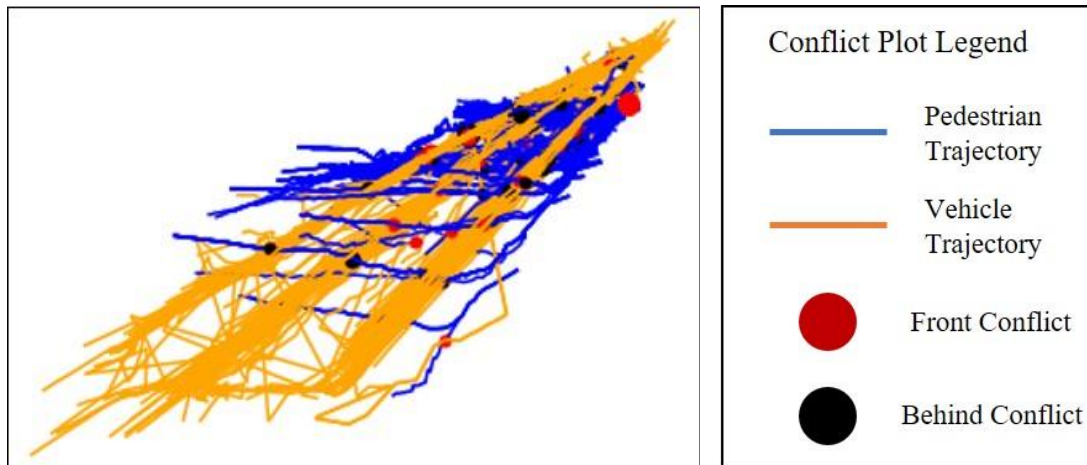


Figure 51 - Conflict Plot, Site 2, Camera 2, Construction Zone

Site 3 contains video at the intersection of M.L.K. and J.E. Lowery for approximately 7 hours. Camera 2’s view shows the limitations of the trailer with the gas station sign obstruction (See Figure 47). In areas where space is limited, finding an ideal view for all four cameras can be difficult.

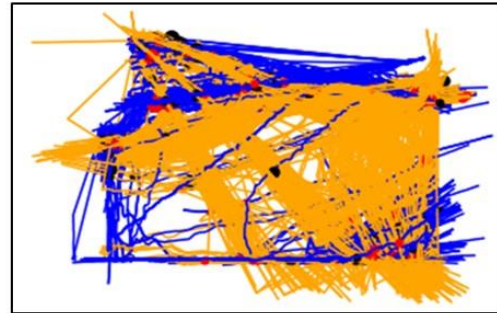
As seen in Table 11 from the conflict analysis output, camera 3 and camera 4 have the most number of conflicts with camera 3 having 721 and camera 4 having 943. Both of these views have more conflicts in 7 hours than Buford Highway had in almost 3 days. This relationship, however, should be expected since cars and pedestrians explicitly interact in an intersection. Surprisingly, both views had more conflicts than behind conflicts, meaning that pedestrians walked ahead of cars. Permissive lefts and rights could cause these relationships. Figure 52 shows both the raw camera 4 view and associated conflict plot. Notice the number of conflicts in the crosswalks.

Table 11 - Conflict Table for Site 3

Camera	Total Conflicts	Behind Conflicts	Behind Near Miss	Conflicts	Near Miss
1	367	213	43	154	140



2	55	35	8	20	14
3	721	356	91	365	302
4	943	443	89	501	446



**Figure 52 - Raw Image and Associated Conflict Plot for Site 3, Camera 4**

Site 4 contained video collected using a GoPro over a period of about a day. This data was collected before any RRFB or HAWK signal was installed at the intersection of 10<sup>th</sup> Street and Myrtle. From the data, over 600 conflicts occurred in a single day. Most of these conflicts were conflicts where pedestrians walked in front of a vehicle. Additionally, a significant number of these conflicts are classified as near misses where a vehicle crosses a pedestrian's path in 3 seconds or less. Figure 53 shows the distribution of conflicts by conflict type, with severe (Sev) meaning the vehicle and pedestrian trajectories intersected within one second of each other.

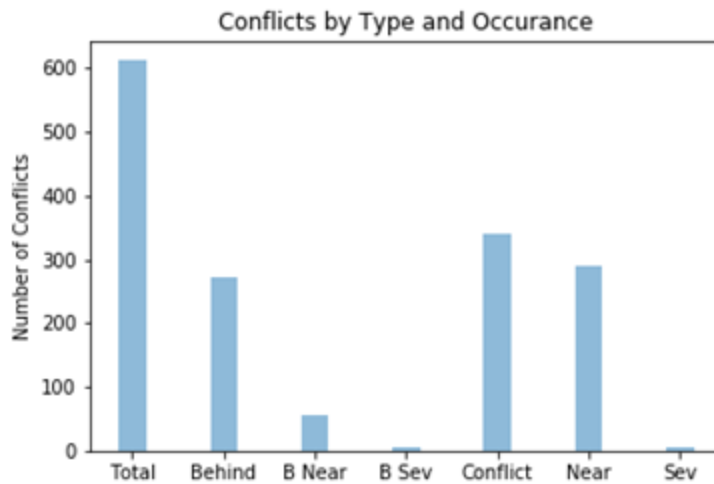


Figure 53 - Conflict Distribution by Type at Site 4

Figure 54 shows that many conflicts occur at the 10<sup>th</sup> Street crossing, as well as the driveway to an apartment and retail complex. Since this time, a HAWK intersection has been installed at this location. The conflict tool can potentially be implemented in before and after studies to understand the effectiveness of conflict mitigation methods.



Figure 54 - Raw Image and Associated Conflict Plot for Site 4

Each site has its own context that a transportation engineer must incorporate in the conflict analysis. For example, Buford Highway and 10<sup>th</sup> and Myrtle are un-signalized sites with right



angle conflicts, while Joseph E. Lowery and Joseph E. Boone are focused on intersections with many conflicts which are likely caused by permissive turns. The conflict analysis allows GDOT to empirically compare different locations and proactively implement conflict mitigation measures.

One method of comparing different sites includes analyzing the exposure rate. Since different sites have different numbers of pedestrians crossing and different context, analyzing the exposure rate provides a normalized comparison of different sites and camera views. Table 12 shows the total number of pedestrians, conflicts, and conflict percentage (total conflicts/total pedestrians) for each site and camera view.

**Table 12 – Normalized Conflict Results**

Site ID	Camera ID	Total Pedestrians	Total Conflicts	Conflict %	View Context
1	1	955	368	39%	Midblock close
1	2	789	336	43%	Midblock far
1	3	749	140	19%	Camera 1 far
1	4	653	664	102%	Camera 2 close
2	1	715	256	36%	Intersection
2	2	314	277	88%	Construction Midblock
2	3	113	14	12%	Midblock Near Intersection
2	4	1243	321	26%	Intersection
3	1	365	367	101%	Midblock
3	2	1055	55	5%	Midblock Near Intersection

3	3	1619	721	45%	Intersection
3	4	2215	943	43%	Intersection
4	1	2879	613	21%	Un-signalized Intersection

Based on the table above, different camera views and sites have varying conflict percentages.

Surprisingly, the number of conflicts on some sites is greater than the actual number of pedestrians. Looking at view context and conflict percentage shows that a relationship between the two exists. The highest three conflict percentages occurred at midblock crossings that were close to an intersection. The lowest two conflict percentages occurred where a midblock crossing would occur; however, an intersection was nearby. From this analysis, proximity to an intersection discourages midblock crossings. Additionally, midblock crossings not near intersections have high conflict rates. This relationship also is intuitive; a person crossing midblock may conflict with multiple cars in each direction. Alternatively, a person crossing at an intersection would likely only conflict with one or two cars that are turning permissively.

This comparison provides GDOT with insights on which areas need infrastructure changes.

Rather than reacting to pedestrian-vehicle crashes or fatalities, the trailer and conflict analysis provide GDOT the data to prevent accidents.

In addition to all analysis-oriented outputs, the conflict code also generates a .csv file that is compatible with the video playback tool. The conflict playback jumps to the time and location where a conflict occurs for a pedestrian. This video playback compatibility allows for ease of manual validation and provides compelling video of people crossing the street. Directions on how to use this tool is included in Appendix 1.

To validate accuracy of the conflict analysis, manual validation was performed. A total of 764 conflicts from the tests above were identified as ‘real’, meaning they were manually viewed in video data using the video playback tool mentioned previously. For each of these 764 conflicts, the pedestrian encroachment times (PET) were logged. The automatically calculated encroachment time was compared to physical time by counting the number of frames in the video between a pedestrian and vehicle trajectory intersecting. Table 13 depicts the average time differences for this dataset of 764 real conflicts.

**Table 13 - Conflict Analysis Validation**

Average Difference (Detection - GT)	Cam1	Cam 2	Cam3	Cam4	<b>Average</b>
All Conflicts	0.14	0.01	0.10	0.11	<b>0.09</b>
All Behind Conflicts	-0.05	-0.07	0.09	0.09	<b>0.00</b>
All Front Conflicts	0.35	0.17	0.11	0.14	<b>0.21</b>
All Away Conflicts	0.36	0.40	0.20	0.27	<b>0.33</b>
All Toward Conflicts	-0.05	-0.08	-0.03	0.00	<b>-0.04</b>
Away & Behind	0.08	0.18	0.16	0.24	<b>0.17</b>
Away & Front	0.81	0.75	0.30	0.33	<b>0.59</b>
Toward & Behind	-0.19	-0.11	-0.07	0.00	<b>-0.09</b>
Toward & Front	0.08	0.00	-0.01	0.00	<b>0.02</b>

Overall, the tracker PET outputs are within a tenth of a second of actual. The main flaw that occurs is associated with away conflicts. Away & Behind conflicts have an average difference of 0.17 seconds, while Away & Front conflicts have an average difference of 0.59 seconds. The location where the trajectories are drawn causes these differences. By default, both vehicle and pedestrian trajectories are drawn at the middle-bottom of the shapes bounding box. For a car

traveling away from the camera, that means the back of the car intersects the pedestrians path, not the front. For cases where a pedestrian walks in front of a vehicle (positive PET), the conflict tool detected outputs on average 0.59 seconds greater than what actually occurred.

In addition to comparing PETs, the manual validation also documented the number of false positives. If a conflict ended up as a false positive, it was categorized into the following different categories: Motorcycle, Turning, Object on car, On Sidewalk, Headlight, and Other. Validation occurred on 1.5 days' worth of data, which had a total initial value of 1508 conflicts. Of these, 1340 were validated as real conflicts. This dataset had an 11% false positive rate. Table 14 contains the validation results from these efforts. The first two rows contain the total accuracies, followed by a breakdown of percent of false positives for each category.

**Table 14 - False Positive Analysis for Conflict Reports**

	<b>Camera 1</b>	<b>Camera 2</b>	<b>Camera 3</b>	<b>Camera 4</b>	<b>Total</b>	<b>Accuracy</b>
<b>Real Conflicts</b>	<b>270</b>	<b>320</b>	<b>128</b>	<b>622</b>	<b>1340</b>	<b>89%</b>
<b>False Positive</b>	<b>98</b>	<b>16</b>	<b>12</b>	<b>42</b>	<b>168</b>	<b>11%</b>
Motorcycle	23	7	7	7	44	26%
Turning	31	0	0	0	31	18%
Object on car	6	1	0	0	7	4%
On Sidewalk	0	1	1	18	20	12%
Headlight	23	6	4	4	37	22%
Other	15	1	0	13	29	17%

## **5) Improve trailer hardware system**

Feedback from GDOT personnel using the system indicated that there was an intermittent issue related to activating the video cameras on the trailer system in the recording tool software. It was

found that there was an IP address conflict in the system related to dynamic IP addressing of the IP interfaces. To fix this, the camera configurations on the trailer system were updated. The IP cameras are connected to the PC using an unmanaged network switch. This resulted in dynamic IP address allocation by Windows as the cameras appear as devices through new interfaces even though the cameras themselves had a static IP address. To make Windows use the IP address as set by the cameras, it needed to be forced onto the same IP address range as the cameras. That is, since the cameras are available on 192.168.1.101 – 192.168.1.104, manually setting IP addresses for those interfaces that connect the cameras to the PC to 192.168.1.201 – 192.168.1.204 (no particular order) fixed the issue.

After evaluating potential improvements to the trailer system such as the ability to run off of generator power, it was determined that any modification would cost too much and not add enough value to the trailer system. This is due to the requirement that any modifications to the current system would require shipping the trailer back to the supplier, incurring a heavy cost. Therefore, this task has been marked complete as it has been overcome by events.

## **6) Perform training and demonstration of the system for GDOT personnel**

The system and software was demonstrated to Jack Anninos and David Jared on August 5, 2019. The new software and user-interface were demonstrated. Discussions for the remainder of the project and close-out plan were also held. Potential follow-on work was discussed, which was included in a needs statement that was submitted in September 2019.

## **Anticipated Impact**

Work performed under this research project will enable a cutting-edge, relatively low-cost platform allowing for rapid classification of pedestrian, cyclist, and vehicle behavior in any area

accessible via video streams. The current state-of-the-art requires custom tailored algorithms, hardware, and contract work incurring heavy costs. Success of this system will enable this data to be collected rapidly and at a low cost. In addition, outputs from the system can allow for automatic detection of “areas of interest” where undesirable or unsafe behavior is occurring. This capability can enhance not only the Georgia Department of Transportation’s capabilities, but those of all transportation authorities both federal and state.

## Conclusions and Recommendations

A new software system based on the Yolov3 neural networking framework was developed and tested allowing for detection and tracking of pedestrians, cyclists, and vehicles in midblock and intersection locations. During the duration of the project, video was acquired and processed for 37 videos over nine separate sites. Table 15 shows results from accuracy testing for each of the classes. Future work should focus on detecting and combining pedestrians that are counted multiple times, resulting in high estimates for the counts, leading to errors.

**Table 15 - Detector and Tracker Accuracy**

Type	Ground Truth	Detected	Detection Accuracy	Count	Count Accuracy
Cyclists	32	29	91%	37	84%
Vehicles	163	158	97%	155	95%
Pedestrians	469	450	96%	546	84%

A conflict analysis software tool was developed and tested allowing for detection of interactions between pedestrians and vehicles. This tool was validated on 13 videos across four different sites. For conflict identification, manual assessment established an accuracy of 89% with a false positive rate of 11% over 1508 potential conflicts. Pedestrian encroachment time automatically

calculated by the system was validated to be within .5 seconds of actual, yielding a high confidence in the system's ability to automatically identify and log conflicts.

A video playback tool was developed that allows for immediate viewing in video of pedestrians, cyclists, and vehicles based on the output from the report generation tool. This tool was extended to allow for viewing of conflicts using the reports generated from the conflict analysis tool.

An approach for localizing trajectories in a geo-spatial frame was developed and tested. This approach worked well for scenes with flat surfaces such as in an intersection or along a flat road. For undulating surfaces, these routines failed. Further work should be conducted in order to enhance the system's ability to geo-locate trajectories in scenes with complex geographies.

## References

1. Redmon, Tamara. "Evaluating pedestrian safety countermeasures." *Public Roads* 74.5 (2011).
2. Chu, Xuehao. "Pedestrian safety at midblock locations." (2006).
3. Cui, Zhenzhong, and Shashi Nambisan. "Methodology for evaluating the safety of midblock pedestrian crossings." *Transportation Research Record: Journal of the Transportation Research Board* 1828 (2003): 75-82.
4. M. Pirkle, *State DOT Looking to Reduce Dangers*, Atlanta, GA: Atlanta Journal Constitution, 2013.
5. Turner, S., et al. "FHWA university course on bicycle and pedestrian transportation." *Draft Report, Federal Highway Administration, McLean, VA* (2004).
6. Li, Bo, Qingming Yao, and Kunfeng Wang. "A review on vision-based pedestrian detection in intelligent transportation systems." *Networking, Sensing and Control (ICNSC), 2012 9th IEEE International Conference on*. IEEE, 2012.
7. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

8. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).
9. Facebook Research. (2020). "Detectron." (Github repository) Facebook. Available online: <https://github.com/facebookresearch/Detectron> last accessed March 2, 2020.
10. NVIDIA. (2020). "NVIDIA Tensor RT." (website) NVIDIA Developer. Available online: <https://developer.nvidia.com/tensorrt> last accessed March 2, 2020.
11. COCO. (2019). "Common Objects in Context." (website) COCO Consortium. Available online: <http://cocodataset.org> last accessed March 2, 2020.
12. Kalman, Rudolf E. "On the general theory of control systems." *Proceedings First International Conference on Automatic Control, Moscow, USSR*. 1960.
13. Wojke, Nicolai, and Alex Bewley. "Deep cosine metric learning for person re-identification." *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2018.
14. Bhardwaj, Romil, et al. "Autocalib: automatic traffic camera calibration at scale." *ACM Transactions on Sensor Networks (TOSN)* 14.3-4 (2018): 19.



## Appendix 1 – Software Manuals

### Pedestrian Video Recording Tool Manual

# Pedestrian Video Recording Tool Software Manual

---

## Description

The pedestrian Video Recording Tool (VRT) is the program required for recording video feeds for future processing for the pedestrian detection system. This program requires that the cameras be positioned in their ideal orientations for recording using a 3<sup>rd</sup> party program as detailed in this document. This tool allows a user to configure the cameras and start recording video in the required format for the Pedestrian Tracker Tool. See the *Pedestrian Tracker Tool Software* manual for details.

## Initial Set-up Process

The trailer system contains a recording PC and four high definition AXIS brand IP cameras. Once the trailer has been physically deployed, a connection to the PC is required in order to configure the cameras and initialize the recording. This is accomplished either by connecting a monitor and a mouse directly to the PC, or by establishing a remote connection to the PC using a windows laptop, the wifi connection, and the remote desktop application provided by windows.

The trailer IP address is 192.168.0.135, you must use this address to establish the connection in the remote desktop software. Please refer to the windows documentation for usage of the remote desktop program. Once a connection is established, the remote desktop program will ask for a password. The password for the system is “trailer”. An image of the Remote Desktop Connection dialog is shown in Figure 55.

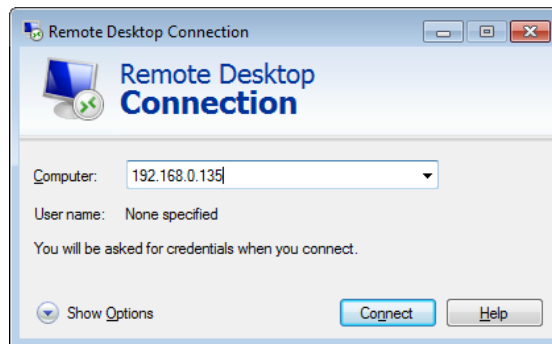


Figure 55 - Remote Desktop Dialog

Trailer IP: 192.168.0.135

Password: trailer

Once the trailer/recording system has been properly deployed, and a remote desktop connection to the PC has been established, the cameras must be positioned using the Axis Camera Companion software. In order to start this software, double click the Axis Camera Companion shortcut on the desktop. An image of this icon is shown in Figure 56.



Figure 56 - Axis Camera Companion Icon

Once the Axis Camera Companion application is run, you will be presented with a password request. There is no password, so this can be skipped by pressing the arrow button to the right of the edit box. This will take you to the main dialog for the cameras. The password request dialog with a red arrow indicating the button to press is shown in Figure 57.

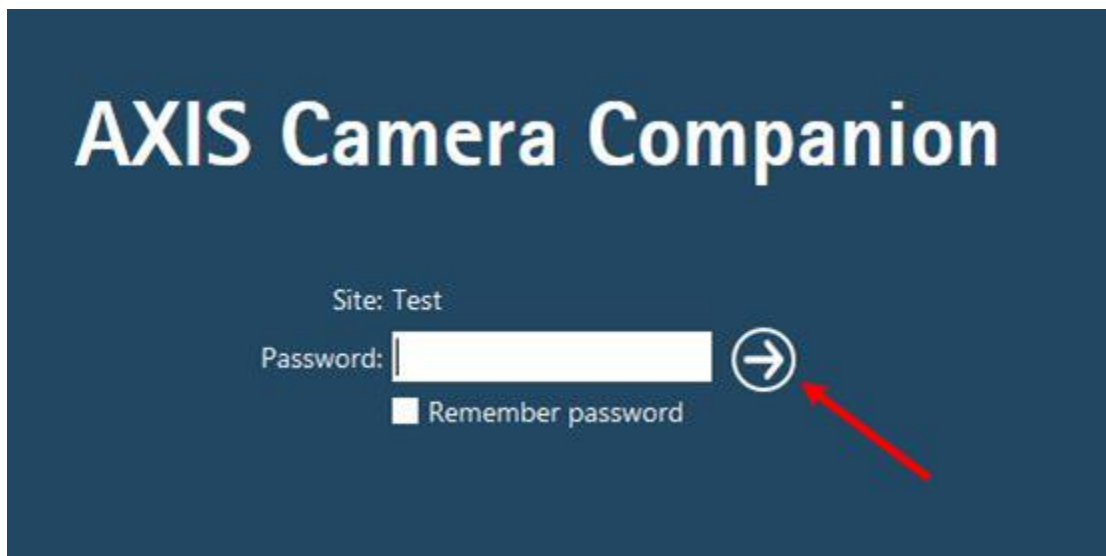


Figure 57 - Axis Camera Companion Login Dialog

Next you will see a display with all four cameras. This is where the cameras can be positioned and zoomed to the appropriate desired locations. Pick a camera by clicking on one of the four camera icons along the bottom of the dialog. Zooming is achieved using the +/- keys on the keyboard or the scroll wheel on the mouse. Changing where the cameras are pointing is simply performed by clicking in the image where you would like the center of the image to be. This will move the cameras to center them on the point that was clicked in the image. A screenshot of this dialog is shown in Figure 58. There is extensive documentation on the operation of this program

located in the Axis folder from the start menu if necessary.

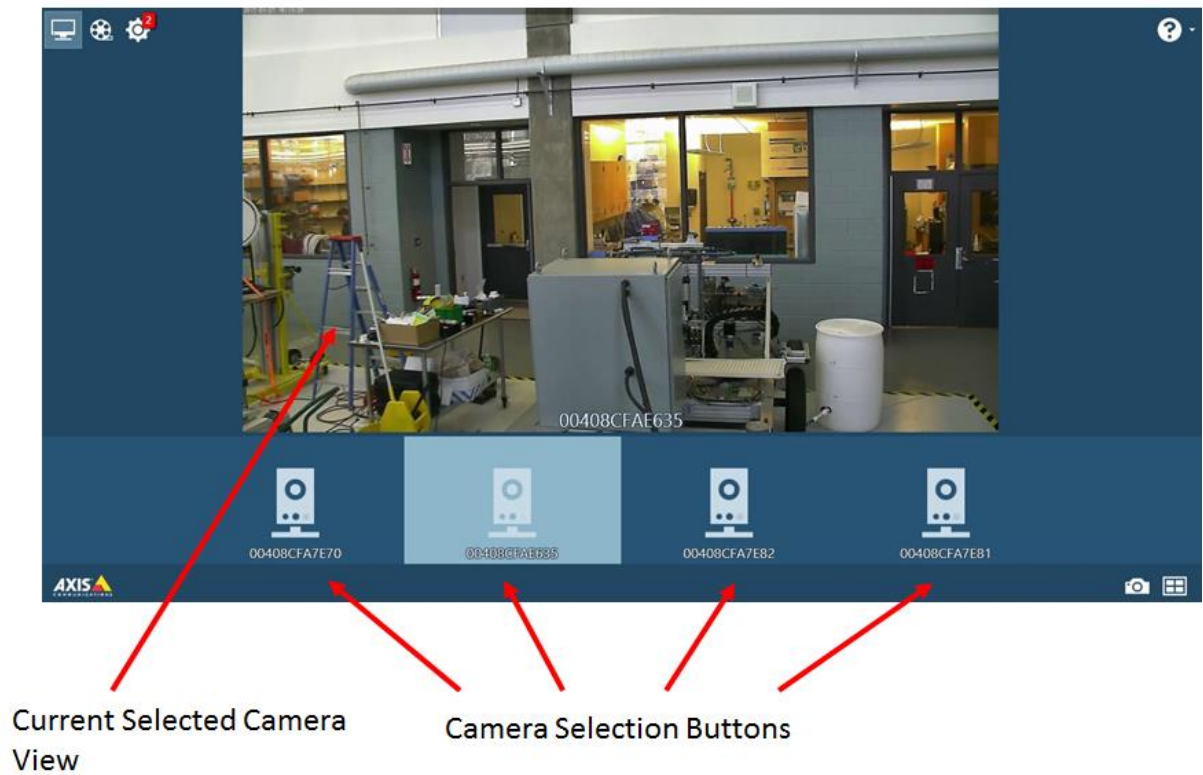


Figure 58 - Camera Selection Dialog

Once all four cameras are positioned appropriately, exit the Axis Camera Companion application and start the VRT. To start the VRT, simply double click the icon on the desktop. The VRT icon is shown in Figure 59.

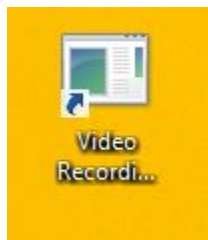
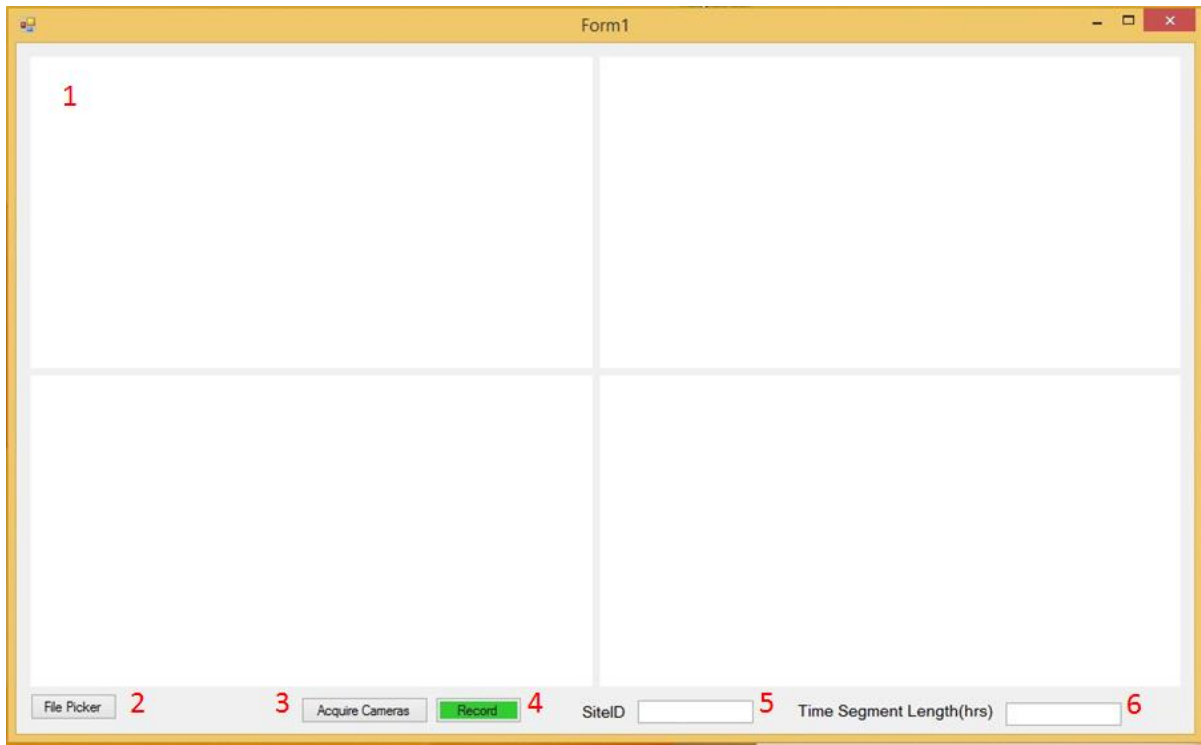


Figure 59 - Video Recording Tool Icon

## VRT – Main Dialog



- |                                    |                         |
|------------------------------------|-------------------------|
| 1 – Camera View Windows            | 5 – Site ID Edit Box    |
| 2 – Save Location Selection Button | 6 – Time Segment Length |
| 3 – Acquire Camera Button          |                         |
| 4 – Record Button                  |                         |

Figure 60 - Video Recording Tool Dialog

### **1 – Camera View Windows**

View for displaying the live camera feeds once the cameras are acquired.

### **2 – Save Location Selection Button**

Button for selecting the folder to save video data.

### **3 – Acquire Camera Button**

Button for acquiring and connecting to all IP cameras.

### **4 – Record Button**

Button to start and stop recording.

### **5 – Site ID Edit Box**

Button to set the site identifier associated with the recorded video.

## 6 – Time Segment Length Edit Box

Edit box for setting the length of time for each individual recorded file in hours. Can accept fractions of an hour. It is suggested not to set this higher than 1 hour.

## Steps for recording Video

This section describes the steps necessary to connect to the IP cameras and record video data.

1. Acquire and connect to the IP cameras using the *Acquire Cameras* button (3)
  - a. A dialog will pop up asking for a password. Because there is no password for the IP cameras, press the “Cancel” button as illustrated in Figure 61 to ignore. You may have to do this once for each camera.

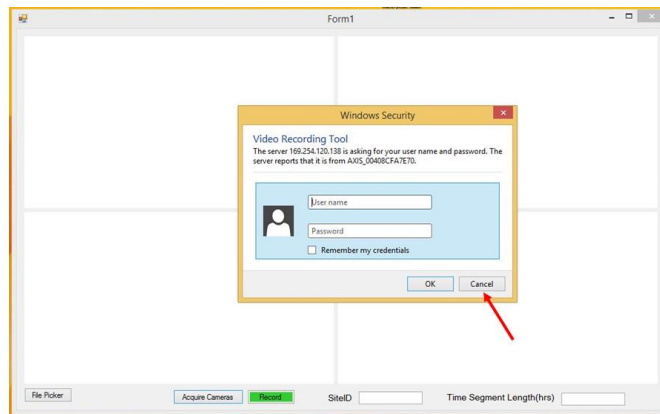


Figure 61 - Video Recording Tool Dialog

2. Press the *Acquire Cameras* button (3) a second time to acquire the cameras. The cameras should appear in the *Camera View Windows* (1) as illustrated in Figure 62. The camera image will be cropped and only a portion will be shown. The full view should be pre-configured via the Axis Camera Companion application as described in the above section.

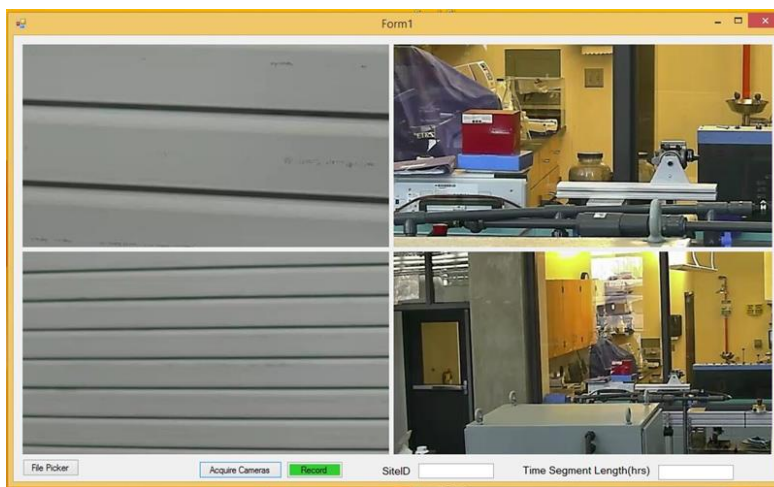


Figure 62 - Video Recording Tool Dialog with Initialized Cameras

3. Select the location to save the images using the *Save Location Selection* button (2). This will open a dialog allowing you to select or create a new directory as shown in Figure 63. This will act as the top-level directory where the files will be saved.

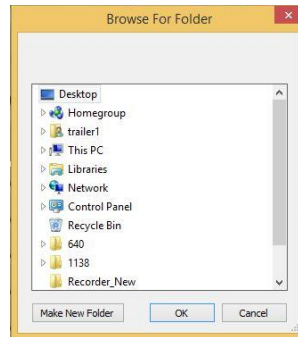


Figure 63 - Save Location Dialog

4. Set the Site Identifier using the *Site Identifier* edit box (5). This will create a subdirectory with the name of the site identifier under the directory that was specified in step 3. All videos will be saved to a subfolder for each camera under the main directory. Do not modify or change the directory names as these are required for the Pedestrian Tracking Tool.
5. Set the time segment length of each video clip using the *Time Segment Length* edit box (6). This will set the length of each recorded video clip. The unit is hours and will take fractions of an hour. It is recommended to use between .5 and 1 hour segments, but any segment length can be selected.
6. Press the *Record* button (4) to initiate recording. When recording the button will change from green to red. You can press the button a second time to stop the recording. The text of the button will change from a green “record” to a red “Stop” when recording as shown in Figure 64.



Figure 64 - Record Button Text

7. When recording, the *Record* button will turn red and read “Stop”. Press the button again to stop the recording. The button will turn green and read “Record” when it has successfully stopped. Recording can be started and stopped as often as desired.

# Pedestrian Tracker Tool Software Manual

## Description

The pedestrian tracker tool is the program for processing raw video data and detecting pedestrians. Their trajectories as they cross the road are recorded to a local database for future analysis. This tool allows a user to specify the video source, configure the image streams, and select which database ID to use for data storage. This manual describes those processes in detail. Figure 65 shows the sample user interface for the tool.

## User Interface – Main Dialog

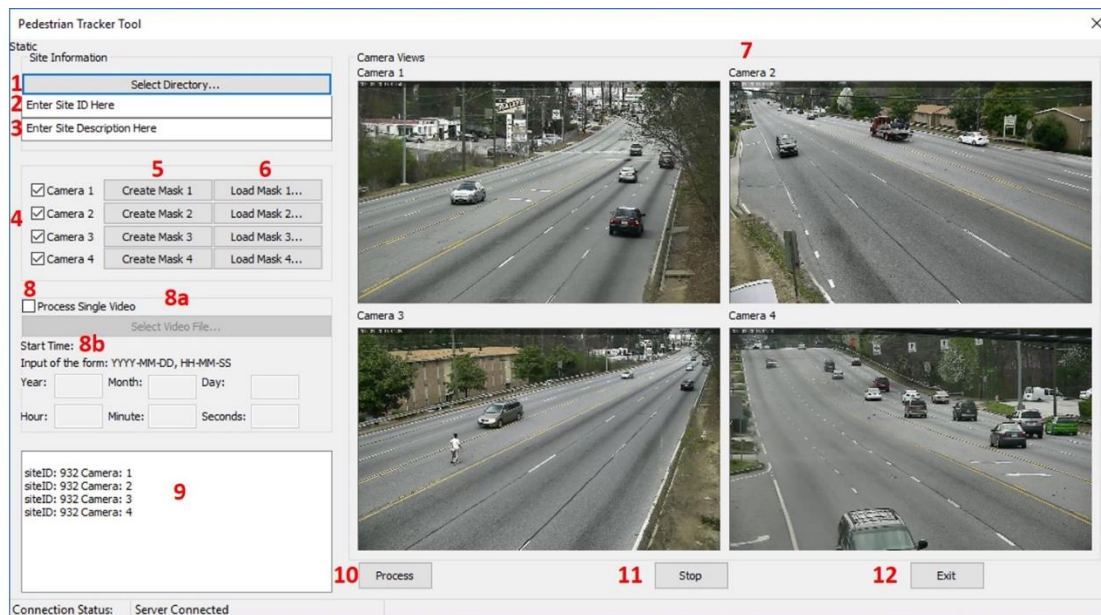


Figure 65 - Pedestrian Tracker Tool User Interface

- |   |   |
|---|---|
| <b>1 – Select Directory Button</b>      | <b>8 – Process Single Video Box</b>     |
| <b>2 – Site ID Edit Box</b>             | <b>8a – Select Video File Button</b>    |
| <b>3 – Additional Notes Edit Box</b>    | <b>8b – Input Start Time Edit Boxes</b> |
| <b>4 – Camera Selection Check Boxes</b> | <b>9 – Process Feedback</b>             |
| <b>5 – Create Mask Button</b>           | <b>10 – Process Button</b>              |
| <b>6 – Load Mask Button</b>             | <b>11 – Stop Button</b>                 |
| <b>7 – Camera View Windows</b>          | <b>12 – Exit Button</b>                 |



## 1 – Select Directory Button

Clicking the *Select Directory Button* will open a dialog asking the user to select the directory containing the raw video data. It is important that the video data is in the format required for processing. This format is automatically created from the Video Recording Tool (see Video Recording Tool Software Document for details). Video files are located in a subfolder named after their site identifier (as assigned in the Video Recording Tool). Each camera has a subfolder containing their respective video files. Please select the top level (Site ID) folder. For example, in the sample file structure in Figure 66, the top level folder that should be selected is named ‘Site1’. Site identifier folder names will typically be numeric (example: 001, 002, 003, etc.).

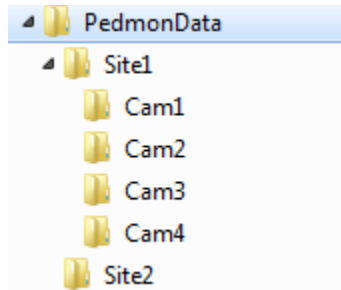


Figure 66 - File Structure Sample

Once selected, the dialog will automatically populate the camera views (7) for the selected cameras with the first image from the first video in each camera subfolder. Only cameras that are selected to be processed will have populated views. The dialog will throw an error if the video file structure is incorrect, or if video files are not located properly.

## 2 – Site ID Edit Box

Enter the Site Identifier number into the *Site ID Edit Box*. This number will be used in the database for all data commitment and future retrieval. This must be the same number as the Site identifier used as the folder name as assigned from the Video Recording Tool.

## 3 – Site Description Edit Box

Enter an optional text description of the site in this edit box. Typically this is used for special notes or for containing the address of the site. This description will be logged to the database in the Site ID table to the Site Description field for future retrieval (See the Database Design Document for details on database tables and fields).

## 4 – Camera Selection Check Boxes

These check boxes will select and de-select which camera videos to process. Only selected (checked) camera video feeds will be processed and results stored in the database. This allows for processing of sites with fewer than four cameras.

## 5 – Create Mask Button

There is a *Create Mask Button* for each camera. This button will only be enabled if the camera is selected via the associated camera check box. This button will open a dialog for creating a new image mask, which is a required step for processing video data. See details and instructions for the Create Mask Dialog in the following section in this document.

## 6 – Load Mask Button

Similar to the *Create Mask Button*, this button is only enabled if the associated camera is selected. This button allows for loading of a previously generated mask using the *Create Mask Button*. This is primarily used for reprocessing of data.

## **7 – Camera View Windows**

Once a site is loaded via the *Select Directory Button*, camera views for each of the selected cameras (via the *Camera Selection Check Boxes*) will be shown in these windows. The image shown is loaded from the first frame in the first available video file in each of the associated camera folders. Once a mask is applied via the *Create Mask* or *Load Mask* buttons, the mask will be drawn onto the image. This allows for visual verification of videos before starting processing.

## **8 – Process Single Video Box**

Checking the *Process Single Video Box* enables both the *Select Video File Button* and *Input Start Time* boxes. Additionally, only Camera 1 in the *Camera Selection Check Boxes* remains enabled. This feature allows for a single video to be processed.

### **8a – Select Video File Button**

The *Select Video File Button* will open a dialog similar to the *Select Directory Button*. Instead of a folder, however, the user selects a single video file to be processed.

### **8b – Input Start Time Edit Boxes**

The *Input Start Time Edit Boxes* requires the user to input the start time of the video. The input start time requires the user to input the start time of the video for the tracking software to put timestamps into the database. If the video start time is unknown, any time can be entered.

## **9 – Process Feedback**

The *Process Feedback* box populates with the Site ID currently being processed. Once the site has completed processing, the *Process Feedback* box will be blank. The “Connection Status” text informs the user whether the server is connected or not. The server must be connected to process results.

## **10 – Process Button**

The *Process Button* will initiate the pedestrian tracking algorithms on the selected video feeds via the *Camera Selection Check Boxes*. Once processing has initiated, tracker results will be logged to the local database with the given Site ID. Current processing time about real time, meaning for each hour of video data, it will take approximately one hour of processing time. This time is faster if fewer cameras are selected for processing.

## **11 – Stop Button**

This button will interrupt processing.

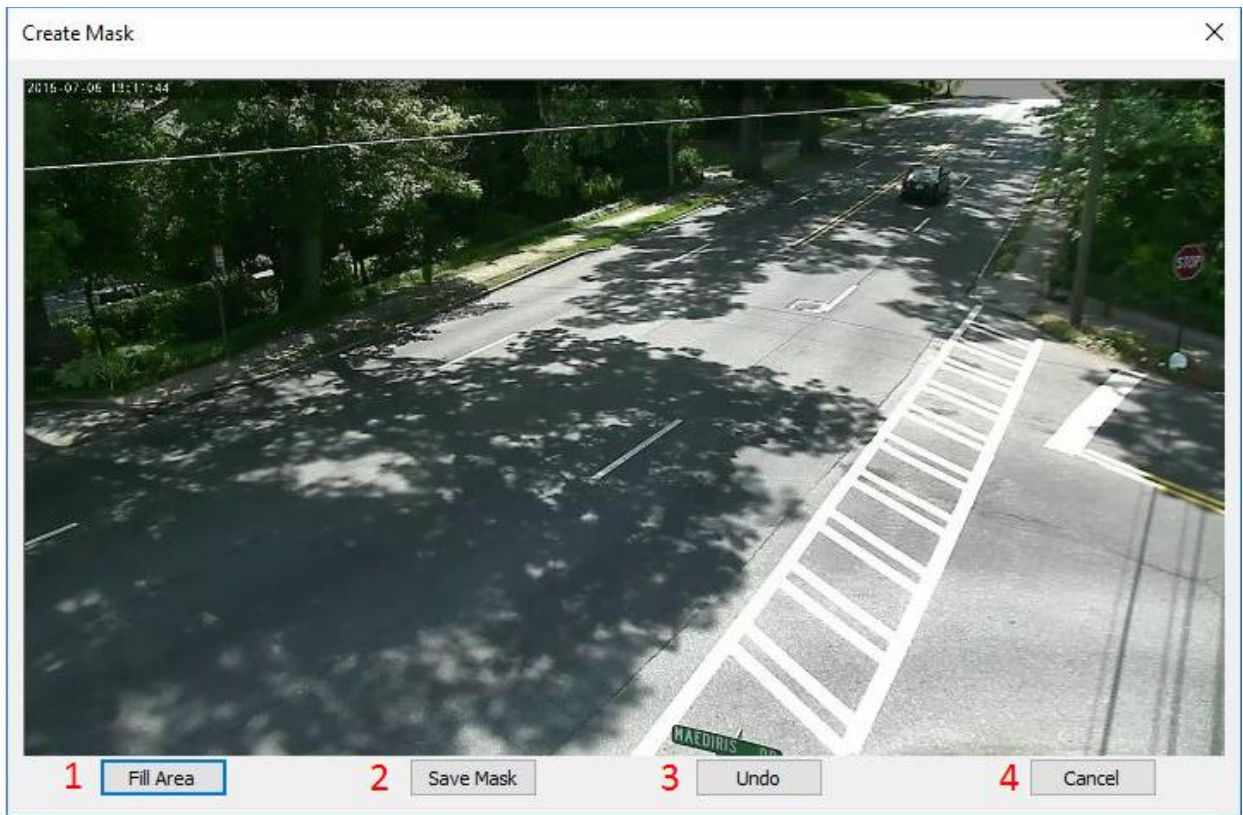
## **12 – Exit Button**

This button will interrupt any ongoing processing and quit the program. If no processing is currently being performed, it will simply quit the program.

## **Create Mask Dialog**

This dialog is created when the user presses the *Create Mask Button* on the main User Interface. This dialog will load the image shown in the corresponding camera view and allow for creation of an image mask. An image mask is simply a way of defining areas in an image to process, and areas to ignore (do not process). This is required to speed up the algorithms and to remove

tracking of trajectories for pedestrians that are not of interest. For example: pedestrians walking on the sidewalk, or crossing in locations that you want to ignore. Figure 67 depicts the *Create Mask* user interface.



- 1 – Fill Area Button
- 2 – Save Mask Button
- 3 – Undo Button
- 4 – Cancel Button

Figure 67 - Mask interface with button labels

### 1 – Fill Area Button

This button will fill the current defined polygon with black pixels (see section Creating a Mask for details). It will be immediately viewable on current image.

### 2 – Save Mask Button

This button will open a save file dialog to save the mask. Type the filename you wish to save as, and click OK. Upon exit, the mask will be saved to disk and automatically applied to the current camera selection for processing. The User Interface will reflect the changes in the mask in the camera view windows. Figure 68 shows masked images ready for processing is shown below. The masked areas are shown as blacked out regions.

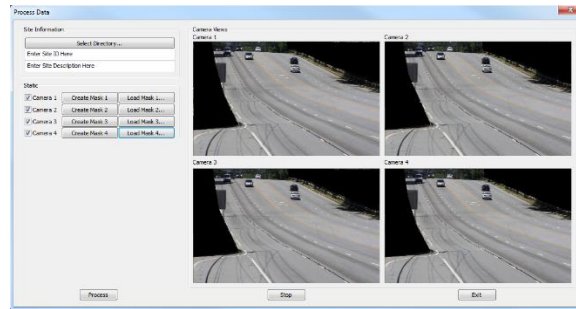


Figure 68 - Sample user interface with masked images

### 3 – Undo Button

This button will remove the last polygon that was filled. Use this if you are not happy with the current mask results. This will only undo the most recent filled polygon.

### 4 – Cancel Button

This button will close the dialog and discard and changes.

## Steps for processing

This section describes the steps necessary to load and process a video.

1. Select the video file directory using the *Select Directory Button* (1)
2. Enter the site ID using the *Site ID Edit Box* (2)
3. Enter the optional site description in the *Site Description Edit Box* (3)
4. Select the Cameras to process with the *Camera Selection Check Boxes* (4)
5. Load or create the image mask using the *Create Mask or Load Mask Buttons* (5,6). See the *Creating a Mask* section for details.
6. Make sure the images are ready for processing by visualizing the masks and images in the *Camera View Windows* (7)
7. Press the *Process Button* (8) to start processing.

## Creating a Mask

This section describes how to create a mask for a camera to prepare for processing. The purpose of creating a mask is foremost to ignore areas of the image where we do not want to track pedestrians such as on sidewalks or other areas that are not of interest for the particular analysis. This section assumes you are already familiar with the dialog and the buttons as described above in this document.

Creating a mask requires filling in polygons in the image with black pixels. The polygons are created by clicking in the image with the left mouse button. 3 or more points defines the polygon for filling with pixels. If you make a mistake, you can always use the *Undo Button*.

- 1) Using the mouse, create anchor points by clicking in the image where you want the corners of your shape to fill.
- 2) Press the *Fill Area Button* (1) to fill the area with black pixels. The dialog will update the image with the filled pixels. Figure 69 depicts a filled polygon.



Figure 69 - Sample Filled Polygon

- 3) Repeat this process as many times as necessary to fill in all the areas of the image that you do not want to process. A finished mask image is shown in Figure 70.

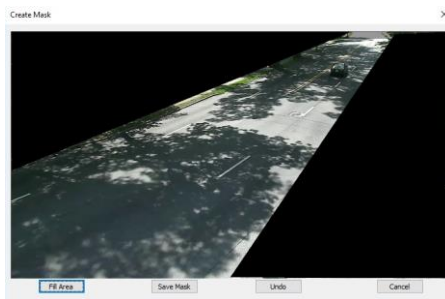


Figure 70 - Sample finished mask image

- 4) Click the *Save Mask Button* (3) to save the mask and apply it to the camera video stream. Once applied it will be immediately visible in the main user interface as shown in Figure 71.

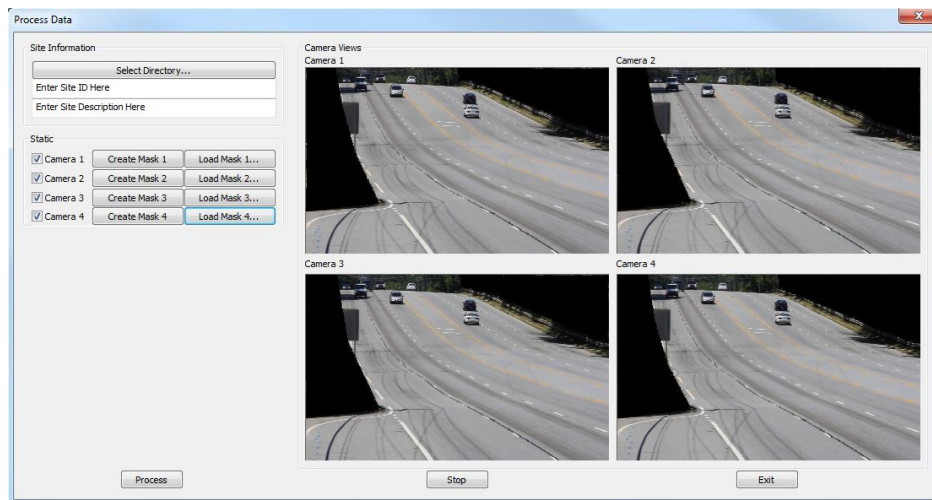


Figure 71 - Sample applied mask

- 5) Click the *Exit Button* if you want to exit the dialog and discard the mask. No image mask will be applied to the video stream. You **MUST** click the *Save Mask Button* to save and apply the mask.

# Pedestrian Report Generation Tool Software Manual

---

## Description

The pedestrian Report Generation Tool is the program for generating and viewing results generated from the pedestrian tracking program. This program is designed to be run after completing analysis with the PedMon Tracker Software. See the *PedMon Tracker Software Manual* for details. This tool allows a user to specify the site identification, select which camera's to query, and select the time period for reporting. This manual describes those processes in detail. Figure 72 shows the sample user interface.

## User Interface – Main Dialog

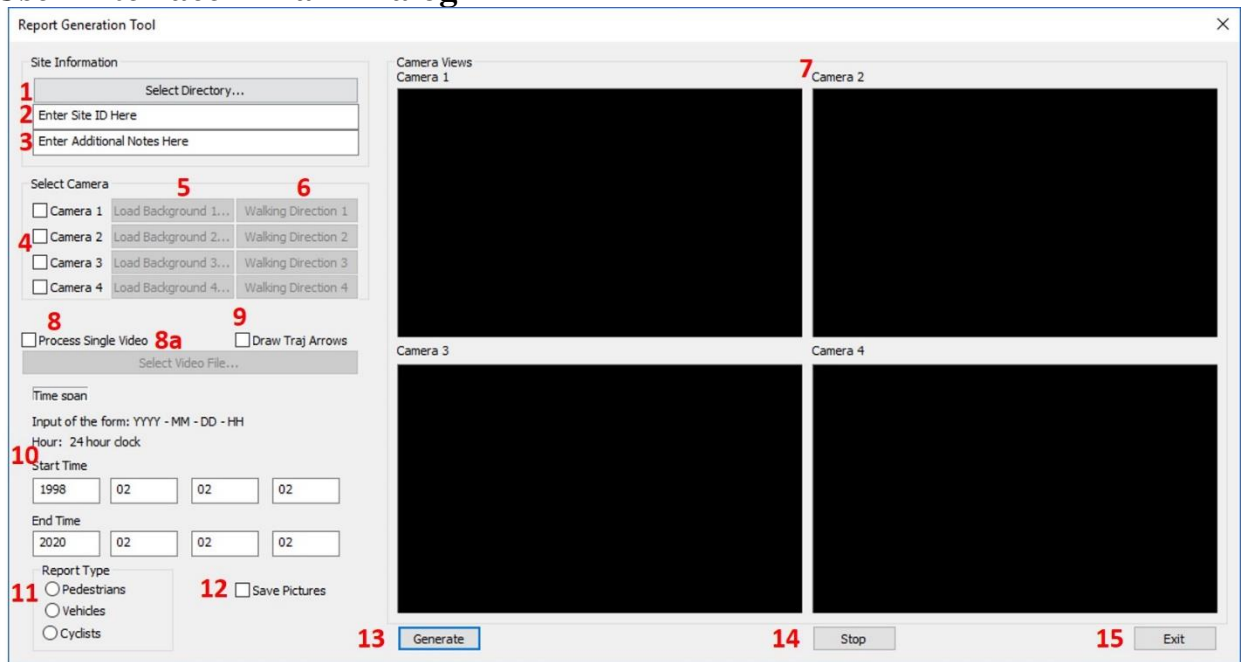


Figure 72 - Report Generation User Interface

- 1 – Select Directory Button
- 2 – Site ID Edit Box
- 3 – Additional Notes Edit Box
- 4 – Camera Selection Check Boxes
- 5 – Load Background Button
- 6 – Walking Direction Button
- 8 – Process Single Video Check Box

- 8a – Select Video File Button
- 9 – Draw Trajectory Arrows Check Box
- 10 – Enter Start/Stop Time Edit Boxes
- 11 – Report Type Buttons
- 12 – Save Pictures Check Box
- 13 – Generate Report Button
- 14 – Stop Button
- 15 – Exit Button



## 1 – Select Directory Button

Clicking the *Select Directory Button* will open a dialog asking the user to select the directory containing the raw video data. It is important that the video data is in the format required for processing. This format is automatically created from the Video Recording Tool (see Video Recording Tool Software Document for details). Video files are located in a subfolder named after their site identifier (as assigned in the Video Recording Tool). Each camera has a subfolder containing their respective video files. Please select the top level (Site ID) folder. For example, in the sample file structure in Figure 73, the top level folder that should be selected is named 'Site1'. Site identifier folder names will typically be numeric (example: 001, 002, 003, etc.).

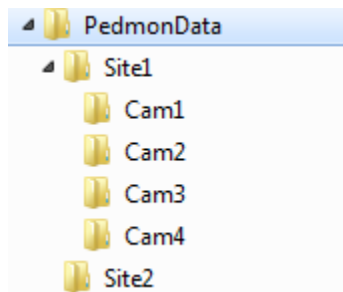


Figure 73 - File Structure Sample

Once selected, the dialog will automatically populate the camera views (7) for the selected cameras with the first image from the first video in each camera subfolder. Only cameras that are selected to be processed will have populated views. The dialog will throw an error if the video file structure is incorrect, or if video files are not located properly.

## 2 – Site ID Edit Box

Enter the Site Identifier number into the *Site ID Edit Box*. This number will be used in the database for all data retrieval. This must be the same number as the Site identifier used as the folder name as assigned from the Video Processing Tool.

## 3 – Additional Notes Edit Box

Use this edit box to enter any notes that may be pertinent to this report. Notes entered here will be printed into the resulting report file generated.

## 4 – Camera Selection Check Boxes

These check boxes will select and de-select which camera trajectories to report. Only selected (checked) camera video feeds will be processed and results generated. This allows report generating for sites with fewer than four cameras.

## 5 – Load Background Button

Report generation requires an image from the site in order to properly visualize pedestrian crossings. If video files are not available as described above using the *Select Directory* button, an alternate method is to load the image manually. This requires that an image from the video data exists. Pressing this button will open a dialog allowing the user to select the image for drawing. The image will immediately show up in the associated camera view window for the selected camera.

## 6 – Select Walking Direction Button



There is a *Select Walking Direction Button* for each camera. This button will only be enabled if the camera is selected via the associated camera check box. This button will open a dialog for selecting the slope of the direction of pedestrian crossing, which is a required step for generating report data. See details and instructions for the Select Walking Direction Dialog in the following section in this document.

## **7 – Camera View Windows**

Once a site is loaded via the *Select Directory Button*, camera views for each of the selected cameras (via the *Camera Selection Check Boxes*) will be shown in these windows. The image shown is loaded from the first frame in the first available video file in each of the associated camera folders. Once the pedestrian crossing direction is applied via the *Select Walking Direction* button, the slope of the crossing will be drawn onto the image. This allows for visual verification videos before starting processing.

## **8 – Process Single Video Check Box**

Checking the *Process Single Video Box* enables both the *Select Video File Button*. Additionally, only Camera 1 in the *Camera Selection Check Boxes* remains enabled. This feature allows for a report to be generated for a single video.

### **8a – Select Video File Button**

The *Select Video File Button* will open a dialog similar to the *Select Directory Button*. Instead of a folder, however, the users selects a single video file to run a report on.

## **9 – Draw Trajectory Arrow Check Box**

Clicking *Draw Trajectory Arrow Check Box* will draw trajectory arrows on the report images for visualization.

## **10 – Enter Start/Stop Time Edit Boxes**

These edit boxes allow the user to enter the start time and stop time for report generation. Pedestrian trajectories will only be reported and visualized for the time period between the start and end time as specified in the start and stop time edit boxes.

## **11 – Report Type Buttons**

The *Report Type Buttons* select which type of report to run. For example, selecting the Pedestrian button will generate a Pedestrian report for the given Site ID. Only one type of report can be run at a time.

## **12 – Save Pictures Check Box**

Clicking the *Save Pictures Check Box* saves individual trajectory images for each unique object. This feature should only be enabled for pedestrians and cyclists.

## **13 – Generate Button**

The *Generate Button* will initiate the report generating algorithms on the selected video feeds via the *Camera Selection Check Boxes*. Once processing has initiated, tracker results will be logged to the local database with the given Site ID. Current processing time is 2-to-1, meaning for each hour of video data, it will take approximately 2 hours of processing time. This time is faster if fewer cameras are selected for processing.

## **14 – Stop Button**

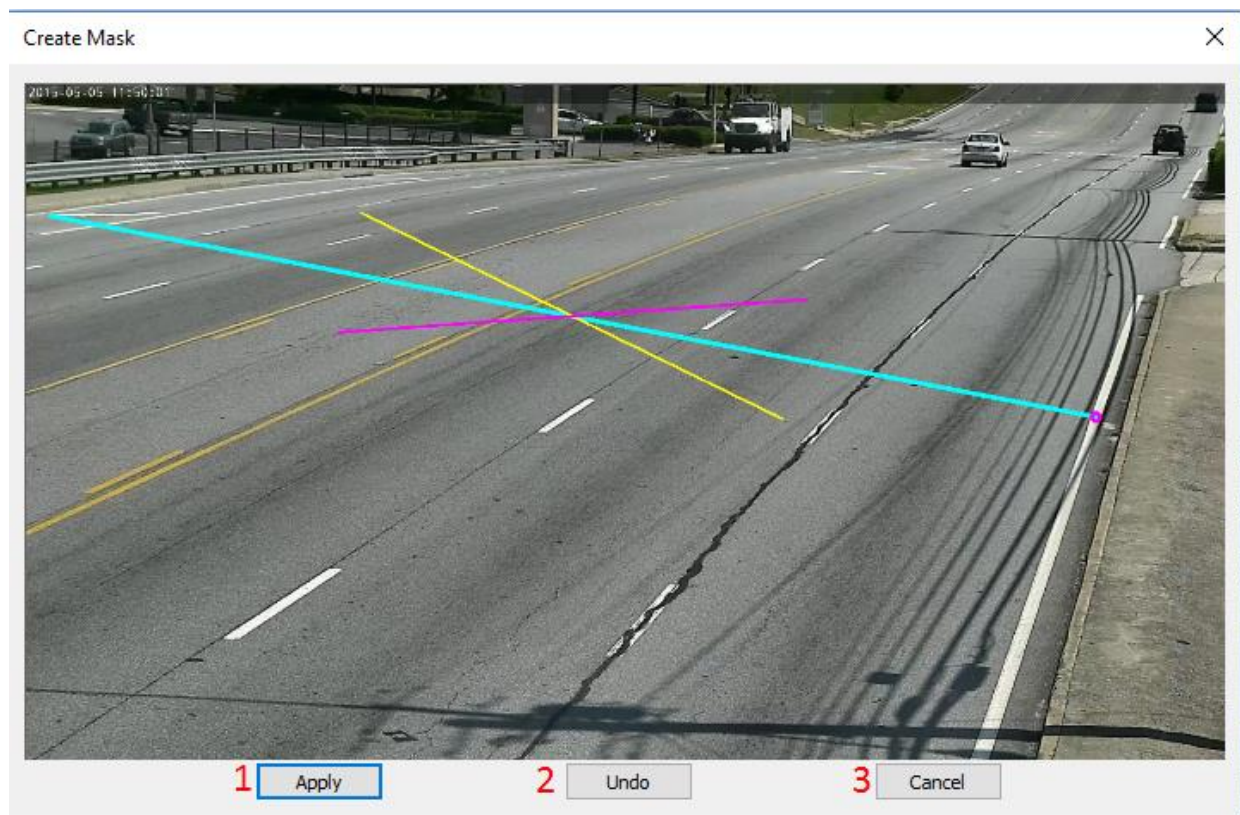
This button will interrupt report generation.

## 15 – Exit Button

This button will interrupt any ongoing report and quit the program. If no generation is currently being performed, it will simply quit the program.

## Select Walking Direction Dialog

This dialog is created when the user presses the *Walking Direction Button (6)* on the main User Interface. This dialog will load the image shown in the corresponding camera view and allow for selection of the direction of travel of pedestrians crossing the road. This is required to filter out invalid trajectories as a result of the tracking algorithms falsely identifying vehicles and tracking them. This is achieved by defining the slope of a line by selecting two points on the road in the direction of pedestrians crossing. Trajectories that are perpendicular to this slope will be rejected for reporting purposes. This effectively filters out the false trajectories due to vehicles. Figure 74 shows the walking direction interface.



- 1 – Apply Button
- 2 – Undo Button
- 3 – Cancel Button

Figure 74 – Select Walking Direction Interface and Buttons

## 1 – Apply Button

This button will fill finalize the slope from the points selected by the user and return to the main dialog. The User Interface will reflect the changes in slope mask in the camera view windows. A sample of the user interface showing images with slopes ready for reporting is shown in Figure 75. The slopes are shown as blue lines in the image.

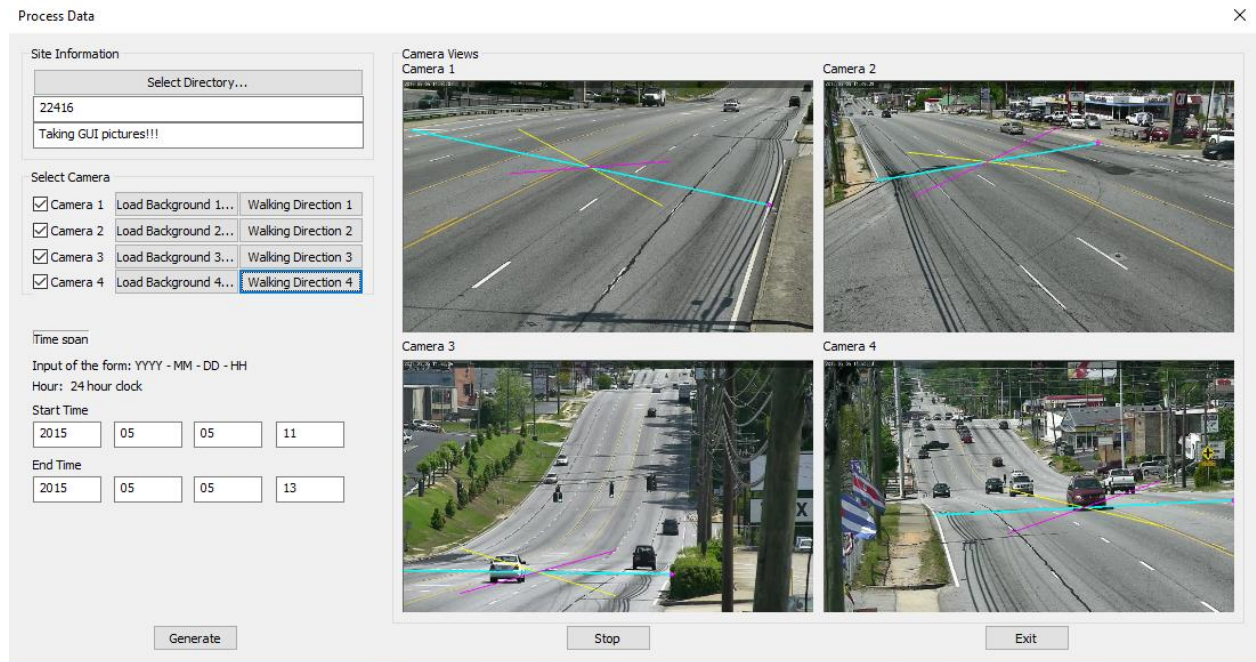


Figure 75 - Sample user interface with slopes

## 2 – Undo Button

This button will remove the last slope that was defined. Use this if you are not happy with the current slope results.

## 3 – Cancel Button

This button will close the dialog and discard and changes.

## Steps for generating a report

This section describes the steps necessary to load and process a video.

1. Select the video file directory using the *Select Directory Button* (1), or alternately, load the individual images using the *Load Background* buttons for each camera (6)
2. Enter the site ID using the *Site ID Edit Box* (2)
3. Select the Cameras to process with the *Camera Selection Check Boxes* (4)
4. Create the pedestrian crossing direction slope using the *Road Direction Button* (5). See the *Select Walking Direction Section* below for details.
5. Make sure the images are ready for processing by visualizing the slopes and images in the *Camera View Windows* (7)
6. Enter the start and stop times in the edit boxes (8)
7. Press the *Generate Button* (9) to start report generation.

## Selecting the Walking Direction

This section describes how to select the pedestrian crossing/walking direction for a camera to prepare for report generation. The purpose of creating a slope for the crossing direction is foremost to filter out incorrect trajectories from vehicles travelling in the direction of traffic. This section assumes you are already familiar with the dialog and the buttons as described above in this document.

Selecting the walking direction requires selecting two points in each image to define the start and stop of a line in the direction of the pedestrian crossing. If you make a mistake, you can always use the *Undo* Button (2).

- 1) Using the mouse, create anchor points by clicking in the image where you want the start and stop points of the line to be.
- 2) Once two points are selected, a line will automatically be drawn into the image indicating the selected crossing direction as shown below. If the slope is incorrect, press the *Undo* button (2) to remove the slope and start over. Figure 76 shows the walking direction.

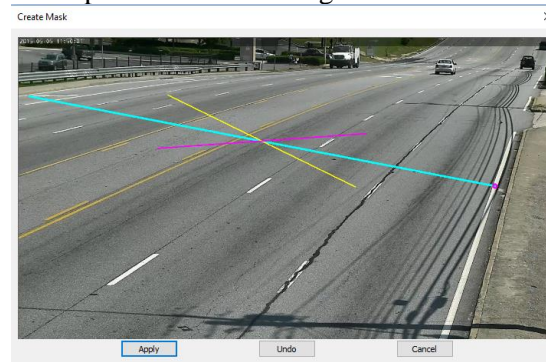


Figure 76 - Drawing walking direction

- 3) Click the *Apply* Button (1) to save the slope and apply it to the camera video stream. Once applied it will be immediately visible in the main user interface in Figure 77.

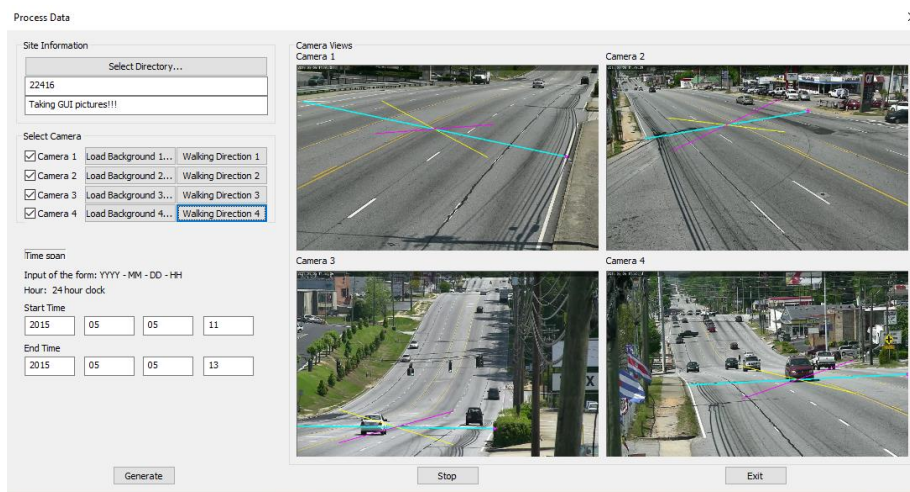


Figure 77 - User interface with walking direction

- 4) Click the *Exit* Button if you want to exit the dialog and discard the selected slope.



# Pedestrian Playback Detection Tool Software Manual

---

## Description

The pedestrian Playback Detection Tool is the program for viewing pedestrian or conflict detection. This program is designed to be run after completing analysis with the Report Generation Tool or Conflict Analysis Tool. See the *Report Generation* and *Conflict Analysis Software Manual* for details. This tool allows a user to specify the video directory, report directory, and jump to detections. This tool allows to jump to a detection directly in the video and watch it. This manual describes those processes in detail. Figure 78 shows the user interface.

## User Interface – Main Dialog

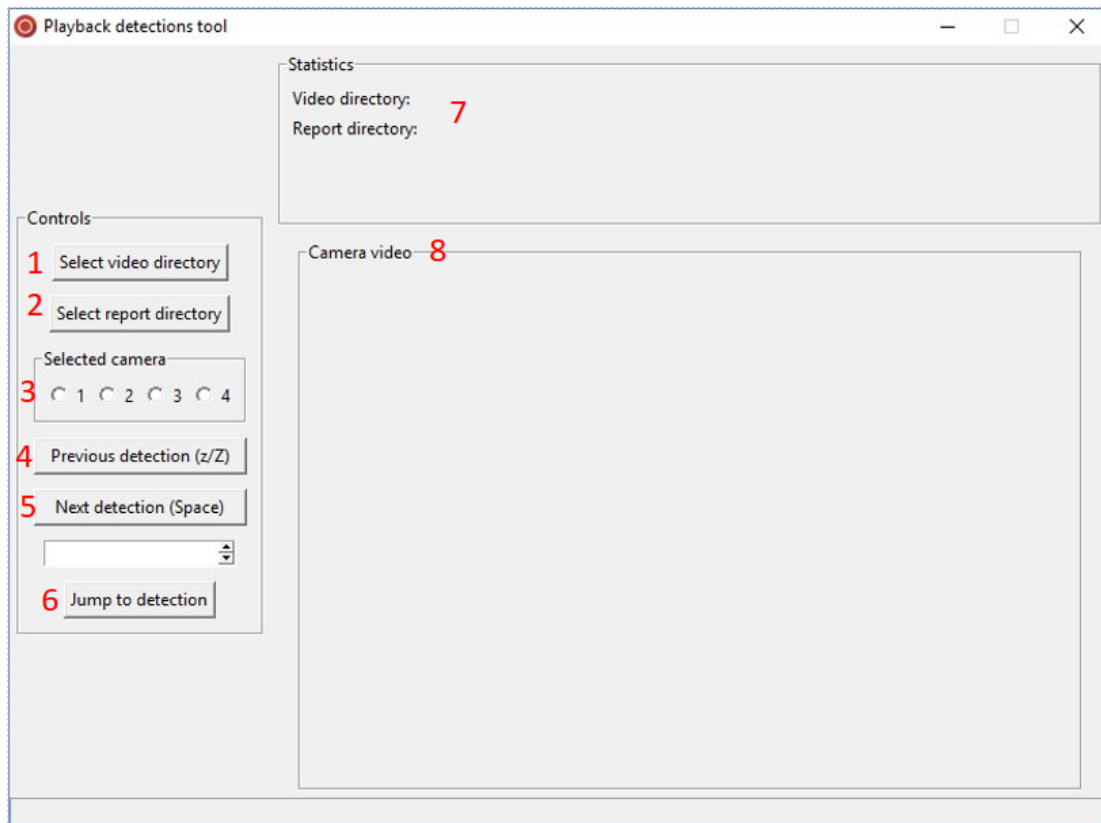


Figure 78 - Playback Detection Interface

- 1-Select Video Directory Button
- 2-Select Report Directory Button
- 3-Selected Camera Button
- 4-Previous Detection Button

- 5-Next Detection Button
- 6-Jump to Detection Button
- 7-Video and Report Directory
- 8-Camera Video Window

## 1 – Select Video Directory

Clicking the *Select Video Button* will open a dialog asking the user to select the directory containing the raw video data. Video files are located in a subfolder named after their site identifier (as assigned in the Video Recording Tool). Each camera has a subfolder containing their respective video files. Please select the top level (Site ID) folder. For example, in the sample file structure in Figure 79, the top-level folder that should be selected is named ‘Site1’. Site identifier folder names will typically be numeric (example: 001, 002, 003, etc.).

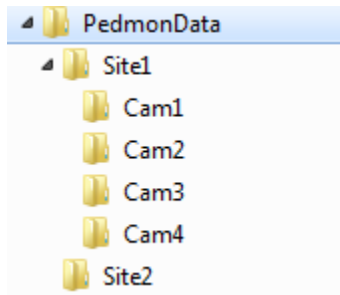


Figure 79 - Select directory

Once selected, the dialog will automatically populate the Video Directory (7) for the selected directory. The dialog will throw an error if the video file structure is incorrect, or if video files are not located properly.

## 2 – Select Report Directory

Clicking the *Select Report Directory Button* will open a dialog asking the user to select the directory containing the report data. The report must be run on the correct video data, otherwise an error will occur. Once selected, the dialog will automatically populate the Report Directory (7) for the selected directory.

## 3 – Selected Camera Button

The Selected Camera Button allows users to choose what camera will be displayed in the Camera View (8). Once clicked, the Camera View (8) will populate with the feed. Figure 80 shows the populated Camera View feed.

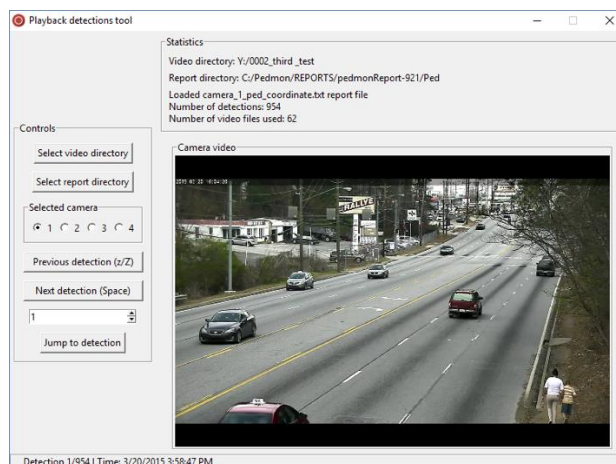


Figure 80 - Playback detection with video

#### **4 – Previous Detection Button**

The Previous Detection Button jumps the previous detection. For example, if the current detection is 2/954, then clicking the Previous Detection Button will jump to 1/954. Additionally, the previous detection feature can be activated by typing “z” or “Z” on the keyboard.

#### **5 – Next Detection Button**

The Next Detection Button jumps forward to the next detection listed. For example, if the current detection is 1/954, clicking the Next Detection Button will jump to 2/954. Additionally, the next detection feature can be activating by hitting the spacebar on the keyboard.

#### **6 – Jump to Detection Button**

The Jump to Detection Button allows users to jump to any possible detection. For example, the current detection is 1/954, but the user wants to jump to 500. Rather than hitting spacebar 500 times, the user types 500 in the input space above (6) and clicks the Jump to Detection Button.

#### **7 – Video and Report Directory**

The Video and Report Directory (7) show where the video and report directories are located, once selected, respectively. Additionally, once the Selected Camera (3) button is clicked, the Number of Detections and Number of Video files used is also displayed.

#### **8 – Camera View**

The Camera View displays the Selected Camera View’s videos being played.

### **Steps for running Playback Detection**

This section describes the steps necessary use the Playback Detection Tool.

1. Select the video file directory using the *Select Video Directory Button* (1).
2. Select the report directory using the *Select Report Directory Button* (2).
3. Select the camera to display using the *Selected Camera Button* (3).
4. Use the *Next Detection Button*, *Previous Detection Button*, and *Jump to Detection Button* to jump through the detections.
5. Select another camera using the *Selected Camera Button* (3) and repeat step 4.



# Pedestrian Conflict Analysis Tool Software Manual

---

## Description

The Conflict Analysis Tool is the program for identifying conflicts from the pedestrian tracking program. This program is designed to be run after the Report Generation Tool. See the *Report Generation Software Manual* for details. This tool allows a user to specify the report folder, and outputs pedestrian-vehicle conflicts and statistics. This manual describes those processes in detail. Figure 81 shows the conflict shortcut which is on the desktop.

## User Interface – Main Dialog



Figure 81 - Conflict tool shortcut

## Conflicts.bat - Shortcut

Double clicking the Conflicts.bat-Shortcut located on the desktop begins the conflict analysis software. Double clicking shortcut opens a console terminal window and prompts the selection of a Report directory.

## Select Report Directory

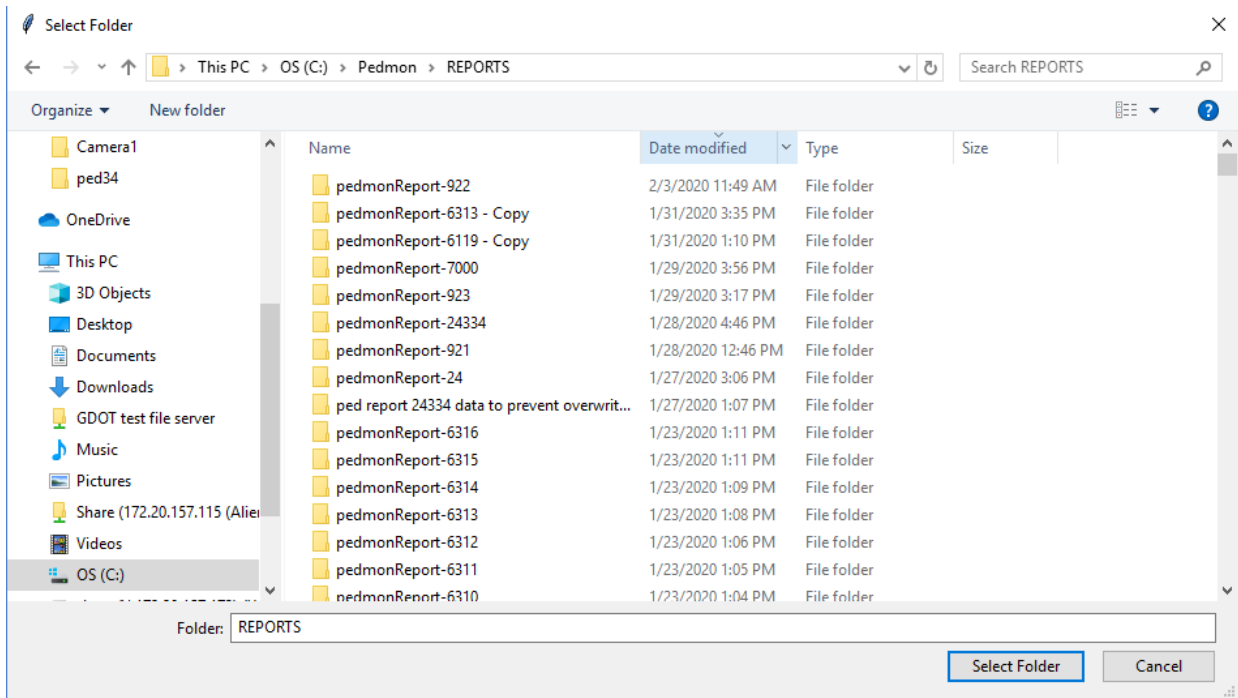
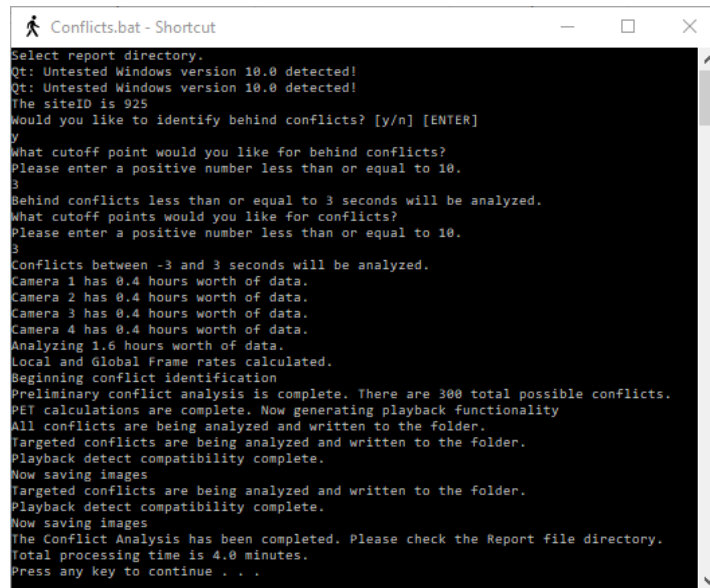


Figure 82 - Select report directory

Figure 82 shows the select the report directory which the conflict analysis will be run on. Once the Report folder is selected, the user will be prompted whether they want to analyze behind conflicts, and the conflict window to analyze.

## Terminal Feedback



```
Conflicts.bat - Shortcut
Select report directory.
Qt: Untasted Windows version 10.0 detected!
Qt: Untasted Windows version 10.0 detected!
The siteID is 925
Would you like to identify behind conflicts? [y/n] [ENTER]
y
What cutoff point would you like for behind conflicts?
Please enter a positive number less than or equal to 10.
3
Behind conflicts less than or equal to 3 seconds will be analyzed.
What cutoff points would you like for conflicts?
Please enter a positive number less than or equal to 10.
3
Conflicts between -3 and 3 seconds will be analyzed.
Camera 1 has 0.4 hours worth of data.
Camera 2 has 0.4 hours worth of data.
Camera 3 has 0.4 hours worth of data.
Camera 4 has 0.4 hours worth of data.
Analyzing 1.6 hours worth of data.
Local and Global Frame rates calculated.
Beginning conflict identification
Preliminary conflict analysis is complete. There are 300 total possible conflicts.
PET calculations are complete. Now generating playback functionality
All conflicts are being analyzed and written to the folder.
Targeted conflicts are being analyzed and written to the folder.
Playback detect compatibility complete.
Now saving images
Targeted conflicts are being analyzed and written to the folder.
Playback detect compatibility complete.
Now saving images
The Conflict Analysis has been completed. Please check the Report file directory.
Total processing time is 4.0 minutes.
Press any key to continue . . .
```

Figure 83 - Terminal feedback

While the conflict analysis is processing, the code provides feedback allows the user to understand what portion of the analysis is currently running. Figure 83 shows sample terminal feedback.

## Steps for running Conflict Analysis

1. Double click the desktop file shortcut seen in Figure 81.
2. Select the report directory to process on.
3. Choose whether you want behind conflicts analyzed or not by typing “y” or “n” and hitting enter.
4. Choose the cutoff times for behind and front conflicts by entering a positive number less than or equal to 10.
5. Let the conflict analysis run. Depending on the length of the report, runtime may be significant. Check terminal feedback to see updates on the conflict process.

The feedback in Figure 83 shows a sample analysis for conflicts between -3 and 3 seconds.

## Conflict Analysis Outputs

The conflict analysis tool creates a Conflicts-all folder in the selected report directory, and the targeted conflicts as well. For example in the figure below, the Conflicts-all folder exists, and the Conflicts--5-5 is the analysis with all conflicts between -5 and 5 seconds. Creating and saving all conflicts allows for faster processing of different time thresholds. Figure 84 shows the sample folder outputs.

Veh	2/7/2020 1:46 PM	File folder
Ped	2/7/2020 1:05 PM	File folder
Conflicts-all	2/24/2020 4:17 PM	File folder
Conflicts--5-5	2/20/2020 1:11 PM	File folder

Figure 84 - Sample folder outputs

Under the Conflict subfolder of the report, folders are created for each camera view. Each camera subfolder has detailed data for each camera view which is seen in Figure 85. Additionally, general conflict data is located in the .txt files in the main subdirectory.

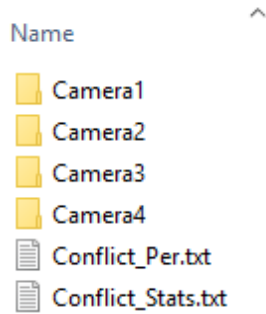


Figure 85 – Sample general conflict outputs

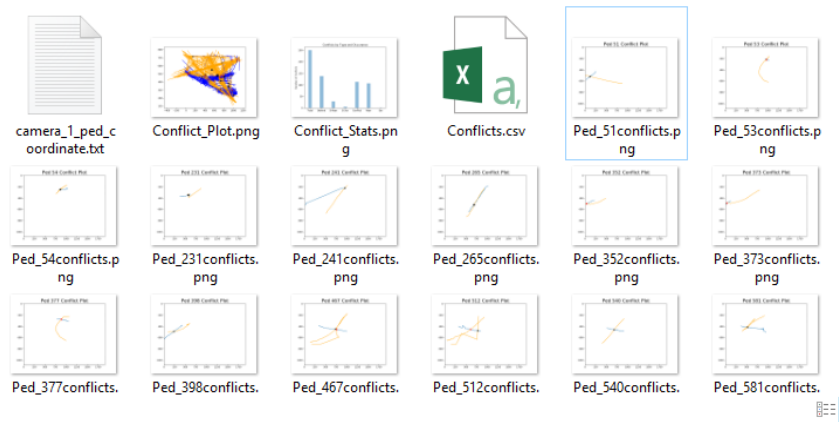


Figure 86 - Sample camera outputs

For each camera, the conflict analysis outputs an overall conflict plot, conflict stats graph, lists all conflicts in a csv file, and has each pedestrians' conflicts saved as an image. Figure 86 depicts a sample layout. Additionally, the coordinate.txt file is used for compatibility with the *Playback Detection Tool*.

## Playback Detection Compatibility

The *Conflict Analysis Tool* has automatic compatibility with the *Playback Detection Tool*. To jump to conflicts in the video, start the *Playback Detection Tool* and follow the user manual. The targeted conflicts folder (Conflicts--5-5) will only depicts conflicts between the cutoff points.

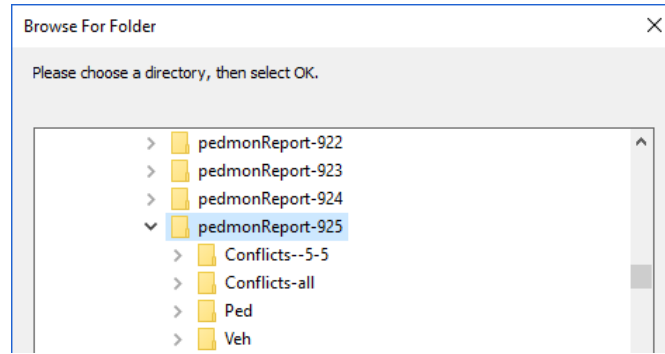


Figure 87 - Playback detection compatibility

When prompted to select a report directory in the *Playback Detection Tool*, select the Conflicts directory, seen in Figure 87, on the desired site. The *Playback Detection Tool* will now display the conflicts in video with the ability to jump to different conflicts. More information about the *Playback Detection Tool* is available in the *Playback Detection Tool Manual*.

## Appendix 2 - Camera Calibration Technical Document

# Camera Calibration for 3D Coordinate Estimation

## 1 - Problem description

Currently, the pedestrian monitoring software looks for potential pedestrians from the camera feed, tracks them in the sequence of images and validates the detections according to some criteria to filter out false positives. This approach has two major deficiencies. One, the detections are done in images directly, without a notion of real world measurements and distances. This is not amenable to heuristics that can take advantage of real world position and speed of the detections, e.g., pedestrians tend to be much slower and smaller than vehicles. But currently, this information cannot be extracted just from the detected boundaries in images. Two, the lack of 3D information does not allow extrapolating the detected image positions to real world coordinates, which might be helpful for surveying. E.g., the object locations in the image cannot be used to determine the GPS location of that detection. Both problems can be addressed by carefully making use of the camera geometry. The goal of this work is to be able to extract 3D position of objects in the camera view as illustrated in Figure 88.

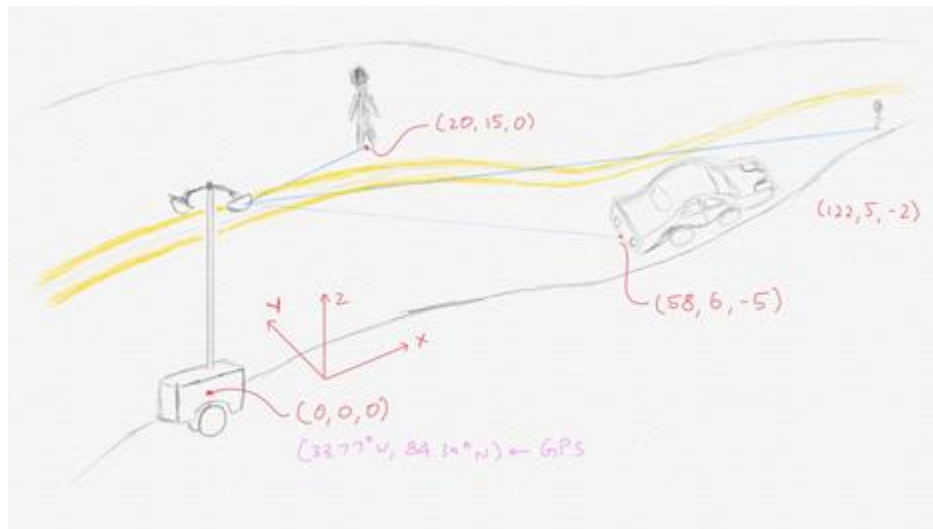


Figure 88 - Concept sketch showing detection of 3D coordinates of different objects on an undulated road from a single camera with respect to the world origin – center of camera wagon. Note all object coordinates could be augmented with global location

## 2 – Camera Properties

In this section, a detailed explanation of properties pertaining to cameras and their geometry is provided. This will be helpful to understand how 3D information can be obtained from monocular cameras.

In practice, it is common to approximate camera models as either a pinhole camera or fisheye a camera. The latter approximates cameras with super wide field of views. Since the cameras that are used in the project operate between 50° to 5° field of view, we can use the pinhole camera approximation.

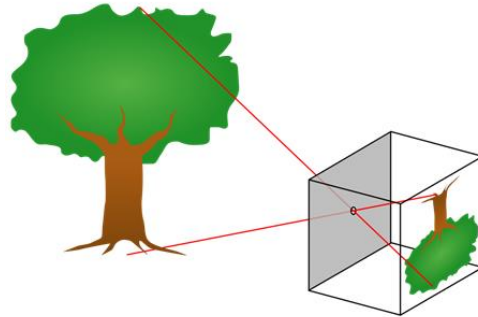


Figure 89 - Pinhole camera model ([https://en.wikipedia.org/wiki/Pinhole\\_camera\\_model](https://en.wikipedia.org/wiki/Pinhole_camera_model))

Figure 89 shows a tree projected on to the back of the pinhole camera, or the wall. This model can be extended to cameras with lens elements and camera sensor – focal length of the lens approximately equals the focal distance (distance between pinhole and the wall) and camera sensor/imager being equivalent to the wall. In general, pinhole camera model entails a projection of the scene on the camera imager according to following relation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} K \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{1}$$

where  $[X,Y,Z]^T$  are coordinates of a point on an object with the center of camera as the origin,  $[x,y,1]^T$  are coordinates of that point projected on the camera sensor virtual plane (note, the “virtual plane” is just a mathematical convenience. In reality, the image forms at  $[-x,-y,1]^T$ .) and  $K$  is often called the camera intrinsic matrix, and captures focal length of camera and the offset of camera sensor. It is given by

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

As it is apparent from 1, the camera projection is a nonlinear transform, solely due to the scaling provided by distance of the object. Intuitively this transformation makes farther objects appear small and nearer objects appear big, with some scaling provided by the focal length. With bigger focal lengths (or more zoom,) farther objects can be scaled up in size and vice versa.

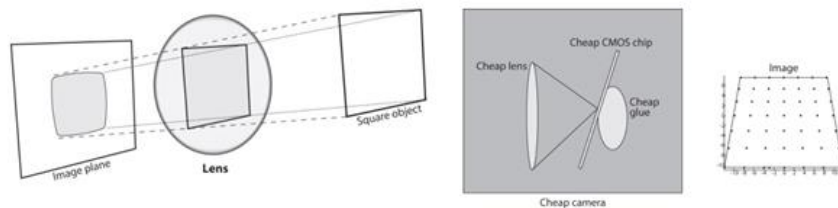


Figure 90 - Two Types of distortion. Courtesy: A. Kaehler, G. Bradski, Learning OpenCV 3

- (a) Radial distortion: rays farther from the center of a simple lens are bent too much compared to
- (b) Tangential distortion results when the rays that pass closer to the center lens is not fully parallel to the image plane (Figure 90).



The pinhole camera model helps finding the points on camera sensor that correspond to points in the world. However, to improve accuracy, it is worthy to resolve for distortions introduced by the camera lens. For the cameras used in the project, two types of distortions can be prominent: radial distortion and tangential distortion. Radial distortion (Figure 90 left side) introduces artifacts that are more prominent in the edges of the images – straight lines tend to bow out. Tangential distortion (Figure 90 right side) on the other hand might result out of inaccuracies in mounting of the camera sensor with respect to the lens.

Due to the distortions, the image coordinates from 1 can be *corrected* by the following relations

$$x_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_1xy + p_2(r^2 + 2x^2) \quad (2)$$

$$y_{corrected} = y(1 + k_1r^2 + k_2r^4 + k_3r^6) + 2p_2xy + p_1(r^2 + 2y^2) \quad (3)$$

where, the first term in 2 and 3 correspond to the correction due to radial distortion with three parameters  $k_1, k_2, k_3$  and the latter terms correspond to that due to tangential with two parameters  $p_1, p_2$ ;  $r$  is the radius from the intersection of optical axis on the imager; and  $[x_{corrected}, y_{corrected}]^T$  being the corrected coordinates for the image coordinates  $[x, y]^T$  from the matrix (1).

Finally, the cameras in the project can be expected to be mounted on a camera wagon or lamp post and it would be more meaningful to see relation between coordinates on the ground with reference to a fixed point on the ground, e.g., base of a lamp post, as opposed to relations with the camera origin. In other words, a fixed world coordinate system is more desirable to reference objects on the ground than the camera itself, which could either move or be present at an inconvenient location to measure from. So, a rigid transformation between camera and world origin can be introduced with the following modification to matrix (1)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{s} \begin{bmatrix} & & \\ & K & \\ & & \end{bmatrix} \begin{bmatrix} R & | & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (4)$$

where  $R$  and  $t$  constitute the rotation and translation respectively that defines the rigid transformation between world and camera origin. Notice that the coordinates in the world is appended with 1, but this is just a notational convenience, to combine  $R, t$  in the same matrix. Also note that scaling factor has changed to  $s$  from  $Z$  in (1).

### 3- Camera Calibration

Calibrating the camera fully to obtain a total of 15 camera parameters can be divided into two stages – Intrinsic calibration and extrinsic calibration.

#### 3.1 – Intrinsic Calibration

On the one hand, typically camera manufacturer's supply some intrinsic calibration information, such as the focal length and sensor offset (if any). For the AXIS Q6045-E cameras used in the project, the focal length range is specified but a one-to-one mapping between *zoom level* and focal length is not. In addition, the sensor size or pixel pitch is not exactly specified, to calculate the focal length in pixels (instead of m). This is important since image coordinates are used in pixels and not any other standard unit. On the other hand, the distortion parameters are rarely

manufacturer specified, except for high precision cameras and since distortion can vary from camera to camera and lens assembly and setting, it would be prudent to calculate the distortion parameters on a case-by-case basis. Hence, all 9 intrinsic calibration parameters must be jointly estimated for the current project.

The calibration marker pattern in Figure 91, printed on to a board, should be shown in front of the camera over several image frames with different orientations. Make sure to impress views from all possible regions of the field of view, at different distances and different tilts. **Note**, this procedure should be done off site with each different camera used in the project, preferably with a selected zoom level. Changing zoom level on site can be adjusted with a look up table for the focal length parameter, but the distortion parameters might change due to presence of multiple lens elements.

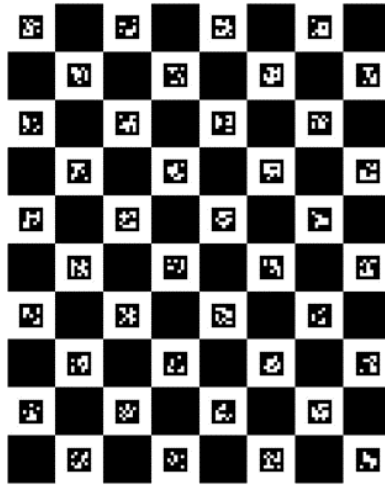


Figure 91 - Calibration Marker Board

The intrinsic parameters are found in two stages. In the first, it is assumed there is no lens distortion and the *homography matrix* for each view is found. The homography matrix arises from the fact that calibration marker pattern is flat and so is the camera sensor. Thus, relating object coordinates of calibration board to image coordinates should be a one-to-one mapping between two planes oriented at two different angles. In other words, 4 modifies to

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{s} \begin{bmatrix} K \end{bmatrix} \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (5)$$

where  $Z = 0$  for all views of the marker board (the object) and  $r_1, r_2$  are the first two columns of rotation matrix  $R$ . This can be further simplified as a homography transformation

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} H \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (6)$$

Note, the homography matrix  $H$  here includes the scaling  $\frac{1}{s}$ . Although  $H$  is  $3 \times 3$  matrix, there are only 8 degrees of freedom for a homography and thus, ideally the matrix per view can be found by using 4 points (or 4 corners of a square) from the calibration pattern. But, more corners – 4 corners from each square – are used to reduce noise (even due to no distortion assumption) and obtain a least squares solution.

Next, to decompose the camera matrix  $K$  and rotation and translation from calibration board to camera origin  $R|t$  from  $H$  as in 5, we will need at least two views since  $H$  will have estimated 8 parameters, but we need to find total 10 parameters – 4 fixed intrinsic parameters ( $f_x, f_y, c_x, c_y$ ) and 6 extrinsic parameters (roll, pitch, yaw and  $t_x, t_y, t_z$ ) that change for each view. Again, as above, more than 2 views – at least, **10** different views – are required to find a least squares solution, that will minimize effects of noise and assumptions. For further details about compiling homography matrices and decomposing them to solve for a camera matrix  $K$  and several rotations and translations can be found in [2].

In the second stage of intrinsic calibration, the distortion parameters are solved. Since the  $f_x, f_y, c_x, c_y$  are obtained from  $K$  matrix, and a bunch of calibration pattern corners in the world in camera coordinates are known, we can estimate the “true” locations of projections as

$$x_{projected} = x_{corrected} = f_x \frac{X_w}{Z_w} + c_x \quad (7)$$

$$y_{projected} = y_{corrected} = f_y \frac{Y_w}{Z_w} + c_y \quad (8)$$

where  $X_w, Y_w, Z_w$  are coordinates of the corners in camera coordinate system. The image coordinates of the corners detected at the start of previous stage  $[x, y]^T$  must correspond to the  $[x_{projected}, y_{projected}]^T$  with slight distortions. Thus, a huge system of equations can be solved to find the distortion parameters in 2 and 3. More details about solving this system of equations can be found in [1]. Finally, the distortion parameters found after solving, viz.,  $k_1, k_2, k_3, p_1, p_2$  can be used to re-solve  $f_x, f_y, c_x, c_y$  in the first stage<sup>1</sup>.

### 3.2 Extrinsic Calibration

Extrinsic calibration involves finding 6 parameters, viz., rotation – roll, pitch and yaw; and translation –  $t_x, t_y, t_z$  that can relate camera’s origin to the “world” origin. For the purposes of this project, origin of the world can be regarded as any fixed, e.g., base of the camera wagon or lamp post, center of the UTM zone, center of the earth etc. For simplicity, the base of the camera wagon – projection of center of the pole on to the ground – is taken as the origin. A larger sized calibration board resembling Figure 91 above needs to be printed out and placed at some location visible to the camera. For wide views, the calibration pattern can be laid flat on the ground, but for very high zooms, it might be worthy to place the calibration pattern perpendicular to the ground with a strong mount. A single image can then be used to calculate the position of the calibration board with respect to the camera accurately, given good intrinsic calibration. That is, after finding all mentioned intrinsic parameters, 4 becomes

$$\begin{bmatrix} x_{corrected} \\ y_{corrected} \end{bmatrix} = \mathcal{U} \left( \frac{1}{s} \begin{bmatrix} & & \\ & K & \\ & & \end{bmatrix} \begin{bmatrix} R & | & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \right) \quad (9)$$

where  $\mathcal{U}()$  is an *undistort* function realizing 2, 3, and  $K$  the camera matrix. The parameters required for the above is found with intrinsic calibration, and so, the parameters that compose  $R$  and  $t$  needs to be calculated.  $\frac{1}{s}$  is directly dependent on the same parameters, and hence doesn't need to be explicitly solved.

Given the corners of each square from the calibration pattern  $[X, Y, Z]^T$ , and the corresponding corrected image coordinates  $[x_{corrected}, y_{corrected}]^T$ , a nonlinear least squares optimization needs to be setup that minimizes the reprojection error. It is nonlinear in nature because parameters of the rotation matrix  $R$  are inherently dependent on nonlinear transformation, for instance trigonometric relations. Reprojection error is the deviation of projected corners of squares from the detected corner points. Note, only a single view of the calibration pattern is required for extrinsic calibration since only 6 parameters are estimated while the number of corner points available from calibration pattern makes system sufficiently overdetermined. Solving this yields the transformation with rotation  ${}^C_B R$ , which signifies rotation from Calibration board to camera; and translation  ${}^C_B t$  from Calibration board to camera. To complete, the reference with respect to world "origin", the transformation between calibration pattern/board to world needs to be specified. In the easiest case, where calibration pattern is laid flat on the ground, the rotation from world to board,  ${}^B_W R$  is the identity matrix  $I$  and the same for translation,  ${}^B_W t$ , will be of the form  $[dist_x, dist_y, 0]^T$ .  $dist_x, dist_y$  specify the distance of the calibration board in length and width from the world "origin", i.e., base of the camera wagon.

## 4 – Road Surface Topology

In a monocular camera, 3D information of the world is projected down to a 2D camera sensor. Therefore, in principle, it is not possible to recover lost information from image coordinates. But, since in this project, we are mostly interested in pedestrians and other objects on the road, we can take advantage of that geometry. Light rays reflecting off road surface, which doesn't necessarily need to be flat, fall on the camera sensor only at a particular location. In other words, there exists a one-to-one mapping between every point on the road visible to the camera (doesn't include occlusions) the camera sensor surface. Although, in practice, the one-to-one mapping doesn't perfectly hold due to quantization, or camera resolution, the correspondences work for the most part. So, if the topology of the road can be determined somehow, 3D locations of detections can be found by inverting the operation performed in 4. The inverted expression, written in a slightly different manner (splitting  $R|t$  matrix,) becomes

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = s \begin{bmatrix} R \end{bmatrix}^T \begin{bmatrix} K \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - \begin{bmatrix} R \end{bmatrix}^T \begin{bmatrix} t \end{bmatrix} \quad (10)$$

It is worthy to note that in 4 describes the projection operation up to a particular scale, which warrants the term  $\frac{1}{s}$ . The constraints introduced by specifying the road topology essentially helps fix this scale. To see this, the simplest topology of the road can be examined – flat. In some

sense, this has already been visited before, in intrinsic calibration. Particularly, 5 describes a homography, relating flat calibration pattern to flat camera sensor. Similarly, if the road is assumed to be flat, with the world origin on the ground, this making all  $Z$  coordinates equal zero, 10 simplifies to

$$\begin{bmatrix} X \\ Y \\ 0 \end{bmatrix} = s \begin{bmatrix} & & \\ & R & \\ & & \end{bmatrix}^T \begin{bmatrix} & & \\ & K & \\ & & \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - \begin{bmatrix} & & \\ & R & \\ & & \end{bmatrix}^T \begin{bmatrix} \\ \\ t \end{bmatrix}$$

which implies, the scaling can be easily found out by using the last equation of this system of equations. That is,

$$\begin{aligned} 0 &= s \left( r_{13} \frac{x - c_x}{f_x} + r_{23} \frac{y - c_y}{f_y} + r_{33} \right) - (r_{13}t_x + r_{23}t_y + r_{33}t_z) \\ s &= f(x, y) = \frac{r_{13} \frac{x - c_x}{f_x} + r_{23} \frac{y - c_y}{f_y} + r_{33}}{r_{13}t_x + r_{23}t_y + r_{33}t_z} \end{aligned} \quad (11)$$

where  $r_{ij}$  are the elements of the  $R$  matrix,  $t_i$  are elements of the translation vector  $t$  and  $f_x, f_y, c_x, c_y$  are elements of the intrinsic calibration matrix  $K$ . Note in the above relations, the terms due to distortions are intentionally left out for brevity. In practice,  $[x, y]^T$  should be equivalent to the “undistorted” image coordinates  $[x_{corrected}, y_{corrected}]^T$ .

In the above example, it is hinted that scaling  $s$  is a function of image coordinates,  $f(x, y)$ . This is true in the general case of arbitrary road topology too. That is, if the road surface is parameterized as an arbitrary continuous function in  $X, Y, Z$ ,  $\text{surf}_1(X, Y, Z) : \mathbb{R}^3 \rightarrow \{0, 1\}$ , where 1 corresponds to presence of surface, then this can be used as additional set of equations to solve for  $s$ , which would be eventually a function of  $x$  and  $y$ . Another, perhaps easier way of parameterizing the general road surface is with a continuous function that maps each  $X, Y$  to a height value. This bears resemblance to a topological map. That is,  $\text{surf}_2(X, Y) : \mathbb{R}^2 \rightarrow \mathbb{R}$ . Then, solving for 3 unknowns  $X, Y$  and  $s$  with 3 equations.

Finally, yet another way to parameterize road surface is with a look-up table. This might be more suitable for arbitrarily complex road surfaces, or for which finding a “surf” function might be hard, but yet maintain a fixed size look-up table. Since, the camera image sensor is quantized into a limited number of (usually square) *pixels*, each ray back traced from the sensor can belong to some quadrilateral projected on to the road surface. Thus, any object (or part of an object) present within this area will appear to be at the same  $[x, y]^T$  location. This implies that if the camera is viewing some scene with a perspective (as opposed to straight on to the road,) rough measurements of the surface topology, farther away from the camera can be forgiving. If some smoothness constraints can be made with this strategy, then the samples of surface measurements can be farther apart, moving away from the camera. An example of this is shown in Figure 92.

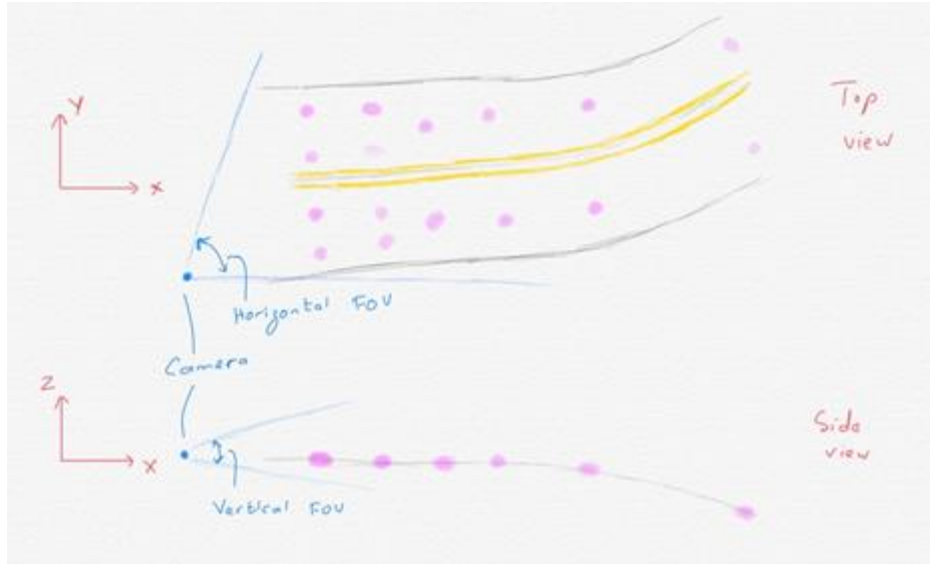


Figure 92 - Sketch of a road surface with a camera viewing down the road. For sufficiently smooth roads, the surface topology can be interpolated from samples (pink), and the samples be made farther apart, farther away from the camera. on to the camera sensor with a set camera calibration, using the “forward” transform.

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} K \end{bmatrix} \begin{bmatrix} R & | & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (12)$$

Note, unlike in 4, the scaling factor  $s$  is estimated for each of the surface samples. A look up table can then be made for each  $x,y$  interpolating the samples obtained in this step, ending up with  $x \times y$  entries. Also note, the look-up table makes the idea that  $s = f(x,y)$  more solid. A potential pitfall to this approach however is that samples from the road surface can be obtained inadvertently that are occluded by the road (such as the samples to the farthest right in Figure 92). However, the look-up table can be adjusted to store interpolated scaling values for those points that satisfy minimum Euclidean distance in camera frame. That is, for each  $[x,y]^T$  in look-up table, store  $\min^p((sx)^2+(sy)^2+s^2)$ .

## 5 – Field testing

Field tests could be done to validate the above derived relations. It is prudent to try field tests at an intersection first since, extrinsic calibration should be easiest for views that are near to camera and also to demonstrate the capability of detecting and tracking traffic/pedestrians from different direction. Some of the steps below are reiterating the procedures explained in sections above.

**Intrinsic calibration** Set the zoom level of the camera off-site. For example, if it is known that the camera wagon will be placed at the corner of an intersection, the zoom level will usually be the lowest setting. Then, after initiating the intrinsic calibration routine, the calibration marker



board must be shown in various different orientations and distances to the camera. It might be helpful to use a monitor to make sure almost all areas of the field of view is covered during this procedure. Figure 93 illustrates a typical sequence of actions needed for this. Finally, the intrinsic calibration software must be signaled the end of the calibration images input, which will result in the software calculating the calibration parameters. If the number of images are not sufficient, the software will indicate. If the zoom level is uncertain for a particular site, the intrinsic calibration could be stored for every zoom level of a camera and retrieved based on the setting chosen on-site.

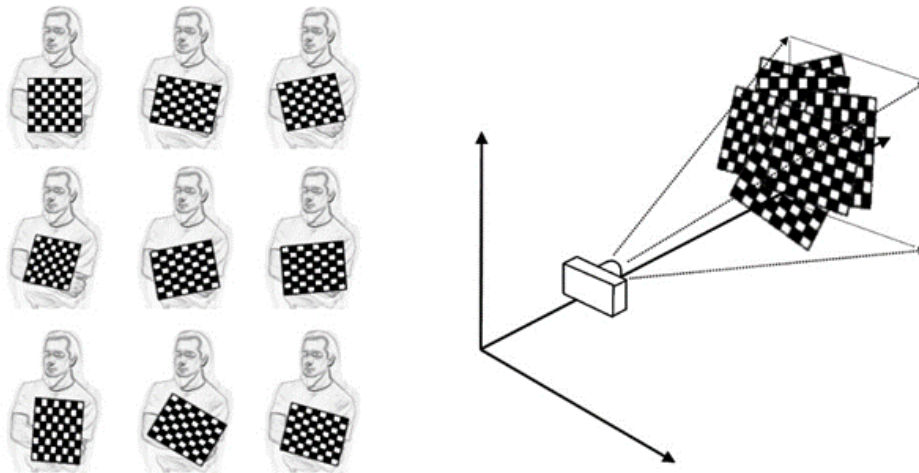


Figure 93 - Typical sequence of images of calibration pattern held at different orientations for intrinsic calibration. Courtesy: A. Kaehler, G. Bradski, Learning OpenCV 3

**Extrinsic calibration** First, establish a “world origin” for the cameras in the field, and make a note of the  $X, Y$  and  $Z$  axes. Typically, the base of the camera pole projected on to the ground would be the easiest. Optionally, take the GPS coordinates (position) and orientation (heading) of this “origin”. Then, setup the camera height and field of view by adjusting tilt and pan controls on-site. Then, place the extrinsic calibration marker pattern flat on the road within view of the camera. Measure the distances  $X$  and  $Y$  of the top-left corner of the marker board to the “world origin”. Initiate the extrinsic calibration software, and feed the position of the marker board. The software must then take an image of the scene and process the camera’s position and orientation with respect to the “world origin”.

**Standard procedure** Start recording videos from the field. With the knowledge of calibration, the rest of the tools developed previously should be able to augment pixel position of objects to 3D information, and if, GPS location of the camera wagon is available, show and plot pedestrian trajectories on a map software or a bird’s eye view visualization of the intersection.

## 6 – Conclusion

In this reading, we have derived a concept to improve the current tools available for pedestrian detection project with 3D information, and hence the capability to find GPS coordinates of each detected object on the road. To accomplish this, the cameras must be calibrated with a two-step process – intrinsic calibration to find parameters specific to each camera, and extrinsic calibration to find location of camera with respect to the road. The former maybe performed only once for each camera, while the latter needs to be performed every time the camera is moved from its site. We have also detailed the procedure to retrieve 3D coordinates of objects on road surfaces with arbitrary topology. Finally, the immediate next steps to validate these procedures in the field has been detailed.

## References

- [1] Brown, D. C. Close-range camera calibration. *PHOTOGRAMMETRIC ENGINEERING* 37, 8 (1971), 855–866.
- [2] Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 11 (Nov 2000), 1330–1334.

## Appendix 3 – Camera Intrinsic Parameters Technical Document

### Camera Intrinsic vs. Focal Length Analysis

This analysis is conducted to see variation of camera intrinsic parameters with various focal lengths. It is also aimed to see the correlation between AXIS Q-6045E API returned *zoom level* and focal length, if any straight relationship exists.

Videos for intrinsic calibration were taken at four *zoom levels*:

- $z_1=162$
- $z_2=1001$
- $z_3=4200$
- $z_4=9999$  (Digital zoom)

Snapshots were taken from each videos and stored in `focal_length_*_images/`. The script `find_intrinsics.py` found intrinsics for four scenarios

In [1]:

```
from utils.config import LoadConfig

# Load calculated params
f1 = LoadConfig('focal_length_1_intrinsics.npz').load()
f2 = LoadConfig('focal_length_2_intrinsics.npz').load()
f3 = LoadConfig('focal_length_3_intrinsics.npz').load()
f4 = LoadConfig('focal_length_4_intrinsics.npz').load()
```

In [2]:

```
# Zoom levels
z1 = 162
z2 = 1001
z3 = 4200
z4 = 9999
```

In [3]:

```
# Average fx and fy
fs = [(f1['camera_matrix'][0, 0] + f1['camera_matrix'][1, 1]) / 2, (f2['camera_matrix'][0, 0] + f2['camera_matrix'][1, 1]) / 2, (f3['camera_matrix'][0, 0] + f3['camera_matrix'][1, 1]) / 2, (f4['camera_matrix'][0, 0] + f4['camera_matrix'][1, 1]) / 2
]
```

```
zs = [z1, z2, z3, z4]
```

In [4]:

```
# Plot focal lengths
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1, 2, figsize=(20, 8))
ax[0].plot(fs)
ax[0].set_title('Calculated focal length in pixel')
ax[1].plot(zs)
ax[1].set_title('Zoom level')
plt.show()
```

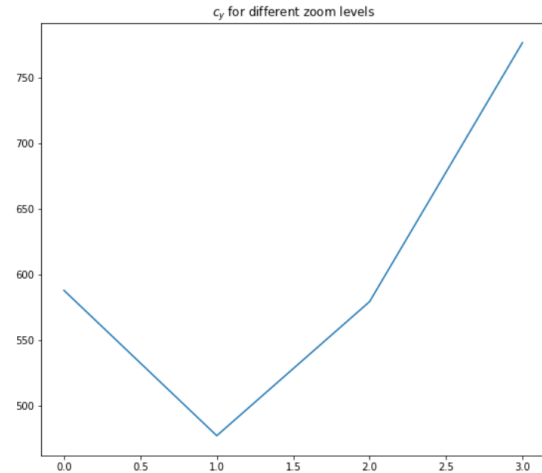
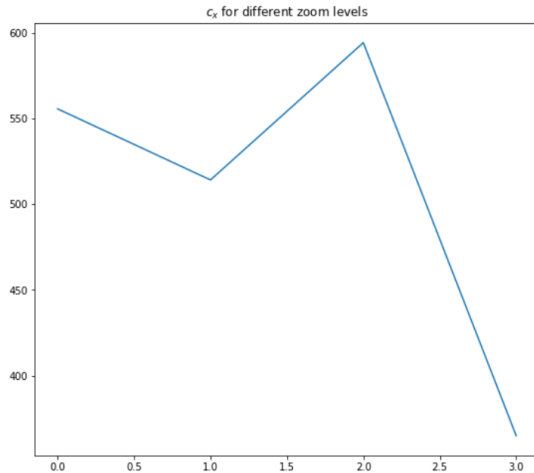
<Figure size 2000x800 with 2 Axes>

The trend is somewhat similar but digital zoom is perhaps causing error with focal length calculation. Let's see principal points for different zooms

In [5]:

```
cxs = [f1['camera_matrix'][0, 2], f2['camera_matrix'][0, 2], f3['camera_matrix'][0, 2], f4['camera_matrix'][0, 2]]
cys = [f1['camera_matrix'][1, 2], f2['camera_matrix'][1, 2], f3['camera_matrix'][1, 2], f4['camera_matrix'][1, 2]]

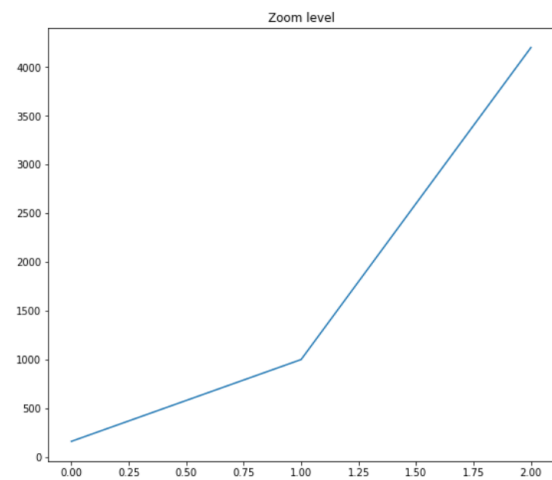
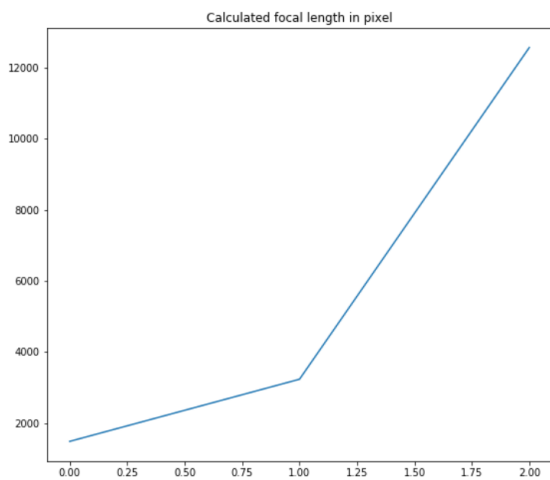
fig, ax = plt.subplots(1, 2, figsize=(20, 8))
ax[0].plot(cxs)
ax[0].set_title('$c_x$ for different zoom levels')
ax[1].plot(cys)
ax[1].set_title('$c_y$ for different zoom levels')
plt.show()
```



Ideally, all principal point calculations should be approximately equal. Unfortunately, digital zoom has completely messed with the last intrinsic calibration. Ignoring the last zoom level then

In [6]:

```
# Plot focal lengths again for first 3 zoom levels
fig, ax = plt.subplots(1, 2, figsize=(20, 8))
ax[0].plot(fs[:3])
ax[0].set_title('Calculated focal length in pixel')
ax[1].plot(zs[:3])
ax[1].set_title('Zoom level')
plt.show()
```



The correlation looks more reasonable. Let's see differences:

In [7]:

```
import numpy as np
diff_f_z = np.array(fs[:3]) - np.array(zs[:3])
print(diff_f_z)
```

The relation looks linear. Let's find the parameters We have three equations of the form  $z_i = s \cdot f_i + o$ , where  $z_i$  is zoom level,  $f_i$  is measured focal length in px,  $s$  is a scale and  $o$  is an offset.

In [8]:

```
zis = np.array(zs[:3])

# x_i = [f_i, 1]^T --> Absorbing the offset into one matrix
xis = np.vstack((np.array(fs[:3]), np.array([1, 1, 1]))).T

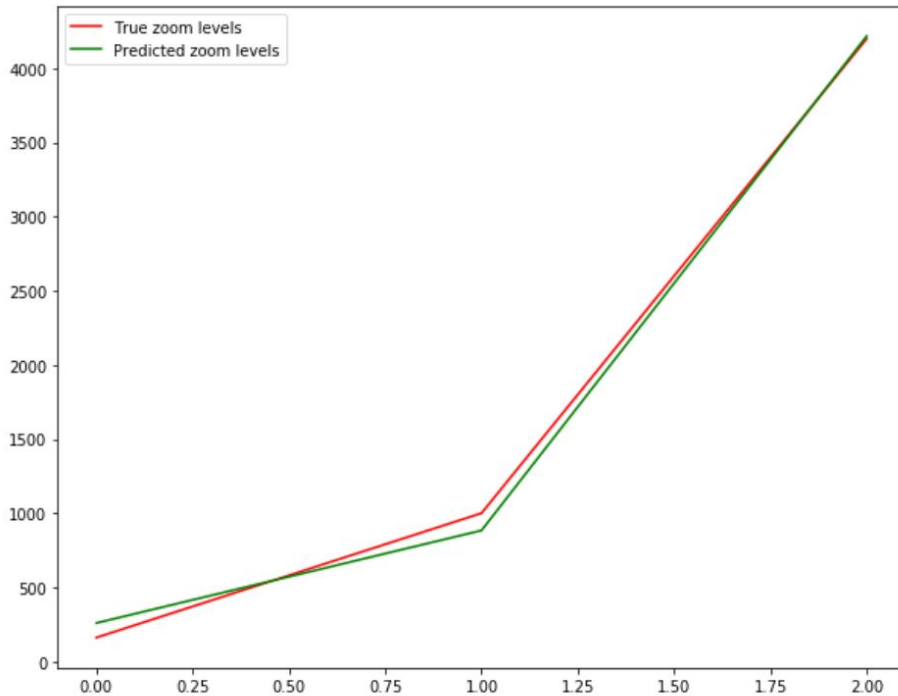
scale, offset = np.dot(np.linalg.pinv(xis), zis)
print('Scale = {}, Offset = {} to go from focal length to zoom level'.format(scale, offset))
```

Scale = 0.3571610356537485, Offset = -269.11190214274563 to go from focal length to zoom level

Let's see how accurate the model is

In [9]:

```
fig, ax = plt.subplots(figsize=(10, 8))
ax.plot(zis, c='r', label='True zoom levels')
ax.plot(np.array(fs[:3]) * scale + offset, c='g', label='Predicted zoom levels')
ax.legend()
plt.show()
```



**Great!** Looks we can take a bunch of calibration from different focal lengths, and obtain a slightly more accurate model for zoom level --> focal length relation. That is,  $f_i = z_i - o_s f_i = z_i - o_s$ . Hopefully the modeled parameters doesn't change between cameras.

Now let's see variation in distortion parameters for different zoom levels

In [10]:

```
k1s = [f1['dist_coeffs'][0, 0], f2['dist_coeffs'][0, 0], f3['dist_coeffs'][0, 0]]
k2s = [f1['dist_coeffs'][0, 1], f2['dist_coeffs'][0, 1], f3['dist_coeffs'][0, 1]]
k3s = [f1['dist_coeffs'][0, 4], f2['dist_coeffs'][0, 4], f3['dist_coeffs'][0, 4]]

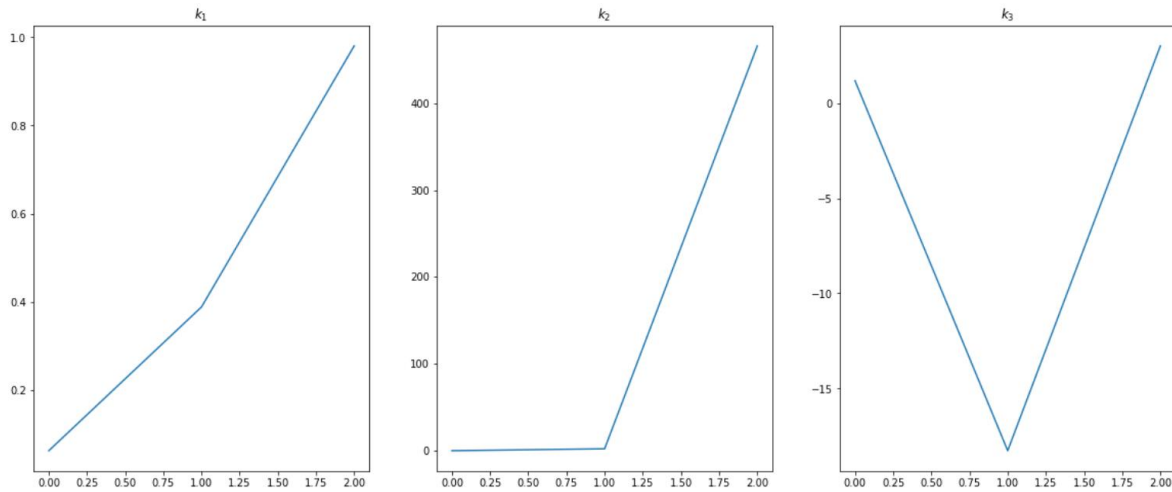
p1s = [f1['dist_coeffs'][0, 2], f2['dist_coeffs'][0, 2], f3['dist_coeffs'][0, 2]]
p2s = [f1['dist_coeffs'][0, 3], f2['dist_coeffs'][0, 3], f3['dist_coeffs'][0, 3]]
```

In [11]:

```
# Radial distortion
fig, ax = plt.subplots(1, 3, figsize=(20, 8))
ax[0].plot(k1s)
ax[0].set_title('$k_1$')
ax[1].plot(k2s)
ax[1].set_title('$k_2$')
```



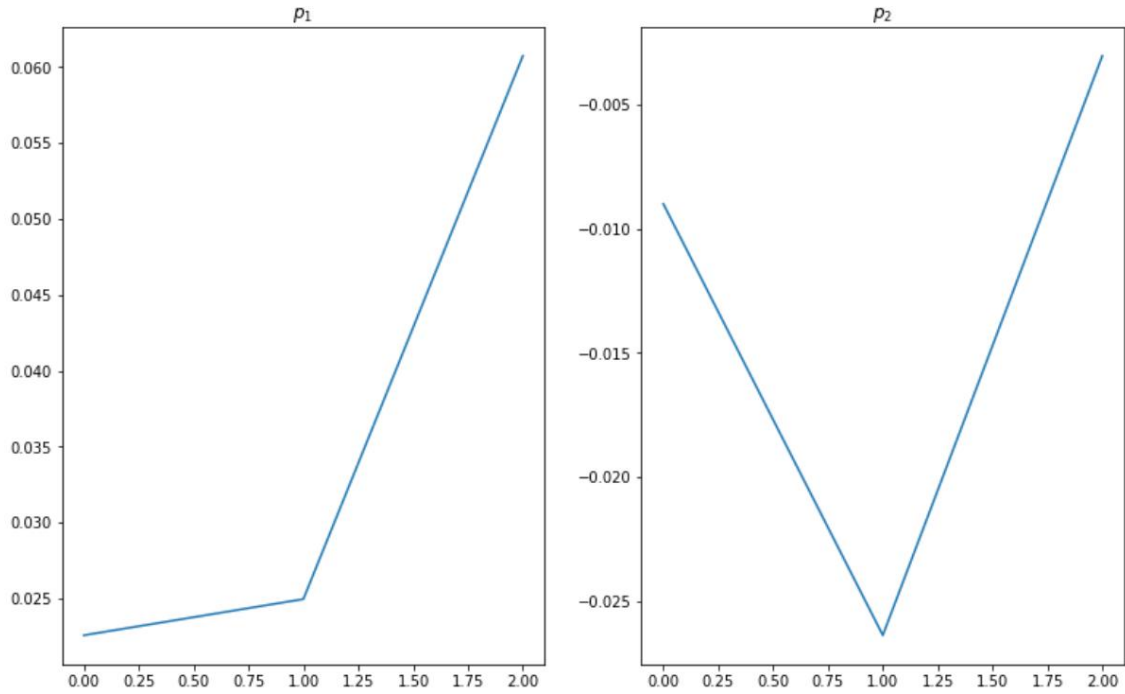
```
ax[2].plot(k3s)
ax[2].set_title('$k_3$')
plt.show()
```



They should ideally be relatively equal for different zoom levels. Unexpectedly, there's huge variation in  $k_2$ .

In [12]:

```
# Tangential distortion
fig, ax = plt.subplots(1, 2, figsize=(20*2//3, 8))
ax[0].plot(p1s)
ax[0].set_title('$p_1$')
ax[1].plot(p2s)
ax[1].set_title('$p_2$')
plt.show()
```



Tangential distortion is OK. These parameters could probably be ignored altogether.

**NOTE: Further investigation required for radial distortion parameters.**

## Conclusion

It appears we can find a meaningful relation between cameras API returned zoom levels and focal length, conditional on gathering more data so that error can be reduced. The variation in radial distortion however needs to be investigated. Particularly, if  $k_2$  has minimal effect on un-distortion.

## Appendix 4 – Literature Review Summary

### Recommendation Section:

For the purposes of pedestrian-vehicle interaction, Post-Encroachment Time will be the best measure. Getting vehicle speed and trajectories will also be important for a more robust measure of pedestrian-vehicle interaction. Midblock specific items to consider would be the pedestrian dwell time on the sidewalk before beginning to cross the road, and whether a pedestrian stops in the middle of the road or uses a rolling gap crossing strategy.

For signalized intersections, additional information to consider would be adherence to traffic laws. Are pedestrians and cars obeying signals? Pedestrian dwell time would also be important to understand pedestrian behavior.

The following literature review includes summaries of studies on automated processing, pedestrian behavior, calibration, and surrogate safety measures.

### Title: Automated Analysis of Pedestrian-Vehicle Conflicts Using Video Data<sup>1</sup>

<https://journals.sagepub.com/doi/pdf/10.3141/2140-05>

#### Summary:

Traffic conflict techniques (TCTs) more important at predicting safety than collision-based measures. Manual data collection expensive and intense. Important Event defined as: “any event that involves a crossing pedestrian and a conflicting vehicle in which there exists a conceivable chain of events that could lead to a collision between these road users”

#### Conflict indicators include:

1. Time-to-collision (TTC) defined as “the time that remains until a collision between two vehicles would have occurred if the collision course and speed difference are maintained”; drawback included the large amount of extrapolation and field work.
2. Post encroachment time (PET): time difference between the moment an offending road user leaves an area of potential collision and the moment of arrival of a conflicted road user possessing the right-of-way
3. Gap Time (GT): variation on PET that is calculated at each instant by projecting the movement of the interacting road users in space and time
4. Deceleration-to-safety time (DST): the necessary deceleration to reach a nonnegative PET value if the movements of the conflicting road users remain unchanged

DST and TTC have underlying assumptions based on vehicle speed that may be inaccurate or unreliable. PET was the most reliable, however, fails to accurately represent a close call with a car having to completely stop. Combination of measures provided most reliable at capturing conflicts.

#### Calibration notes:

Calibration composed of 22 points form selected salient features in the traffic scene.

The world coordinates of the calibration points were collected from an orthographic image of the location obtained from Google Maps. The intrinsic parameter considered in this study is the camera focal length. The mapping in Equation 1 imposes a reduction in dimensionality due to the projection onto a plane. The inverse projection is defined only if one of the world coordinates, or a relationship thereof, is known. In the current application, image plane coordinates are re-projected onto the road surface, that is, the plane  $Z = 0$ .

Similar studies used artificial construction of an orthographic image by using video image rectification. Used feature tracking and grouping. Two important parameters  $D(\text{connection})$  and  $D(\text{segmentation})$ . Used overhead video, so would likely be easier to project

Algorithm for Calculating Conflict Indicators for Pedestrian–Vehicle Event: section which has the formulas to calculate multiple indicators.

Some issues with TTC because severe events may not be on a “collision” course. TTC overestimates conflicts. PET was most reliable parameter, but has some inherent drawbacks with severity (ie, when a vehicle comes to a complete stop, then goes shortly after a pedestrian would walk)

**Title: The development of an automatic method of safety monitoring at Pelican crossings<sup>2</sup>**

<https://doi.org/10.1016/j.aap.2005.04.012>

Summary: Most driver non-compliance occurs during flashing amber period, and other key conflicts were with pedestrians and the driver green phase likely due to delays pedestrians were experiencing. To avoid conflict most drivers would decelerate while some would decelerate and swerve.

Results show that deceleration rates are a valid safety indicator. Deceleration of  $6 \text{ m/s}^2$  is deemed a serious conflict. Deceleration of  $4.5 \text{ m/s}^2$  is a slight severity of conflict. And deceleration of  $3 \text{ m/s}^2$  is a potential conflict. These deceleration rates correspond to time to accident of 1.6, 1.8, and 2, respectively. Used loop detectors to get speed of vehicles and combine it with pedestrian data.

Calibration and Trajectory Data: Data from video transcribed using VIDS and PROGRESS programs. Used pneumatic loops in 3 different configurations to get vehicle speed and deceleration rates.

Used formerly logged deceleration data, while also having multiple different loops set up

**Title: Automated safety diagnosis of vehicle–bicycle interactions using computer vision analysis<sup>3</sup>**

<https://www.sciencedirect.com/science/article/pii/S0925753513001240#b0150>

Summary: Discusses automated analysis for bike-vehicle conflicts at a troubled intersection. Used TTC as the vehicle conflict measure. They used TTC in a probabilistic terms rather than deterministic and also considered events that had a collision course for part of the time versus all the time. Cut off of TTC min 3 s or less.

Calibration and Tracking Methodology: Used Kanade–Lucas–Tomasi Feature Tracker algorithm. Each calibration process begins with the user annotating features in the camera image and in an aerial, orthographic image of the intersection.

**Title: Automated Detection of Spatial Traffic Violations through use of Video Sensors<sup>4</sup>**

<https://journals.sagepub.com/doi/abs/10.3141/2241-10>

Summary: Able to detect certain traffic violations automatically using video detection. Could be used as a surrogate measure for safety.

Calibration and Tracking: Used k-means clustering and pattern matching using the longest common sequence similarity measure (LCSS). Used same calibration procedure as a previous study they conducted.

**Title: Automated video analysis as a tool for analyzing road user behavior<sup>5</sup>**

<https://portal.research.lu.se/portal/files/5708522/1032713.pdf>

Summary:

Distances between objects from camera is difficult. Camera high above an intersection is an ideal location, but impractical. Cameras could be mounted on buildings and users can be projected using rectification. Also discusses using two cameras and an epipolar line to transform 3D coordinates into 2D.

Overestimation of derivative values is a problem. “To overcome these problems we used local kernel polynomial regression algorithm ([6]). One advantage of this method is that the size of the bandwidth is an adjustable parameter, calculated dynamically depending on the data character in the current region. The other advantage is that the method allows separate estimation of the derivatives directly from the raw data.”

Also discusses speed estimation using displacement between frames. “To do that the image is interpreted as a sampled version of a two dimensional function. According to the sampling theorem it is possible to reconstruct the original function as it was before the sampling by using sinc-interpolation between the sample points (pixels). This assumes that the original function contains no frequencies higher than half the sampling frequency, which is usually guaranteed by the optical properties of the camera lens.”

Based on a previous study, jerk may be a better indicator of conflict severity than acceleration / deceleration rates.

**Title: Analyses of pedestrian behavior on mid-block unsignalized crosswalk comparing Chinese and German cases<sup>6</sup>**

<https://journals.sagepub.com/doi/full/10.1177/1687814015610468>

Summary: Pedestrian behavior in China and Germany is significantly different. In Germany, most pedestrians will begin crossing with the first car gap they see, exact opposite in China. Most wait for the entire platoon to finish before crossing.

**Title: An accident waiting to happen: a spatial approach to proactive pedestrian planning<sup>7</sup>**

<https://www.sciencedirect.com/science/article/pii/S0001457502001495>

Summary: Research supports Risk Homeostasis Theory where people target a level of risk that maximizes difference between perceived benefits and costs of a choice. Differences exist between perceived safety and number of crashes for UNC Chapel Hill.

**Title: Features of Pedestrian Behavior in Car-to-Pedestrian Contact Situations in Near-Miss Incidents in Japan<sup>8</sup>**

<https://doi.org/10.1080/15389588.2013.796372>

Summary: Difference exists between Vehicle TTC and pedestrian TTV (time to vehicle). TTC depends on the amount of obstruction to the pedestrian. The shortest was when pedestrians moved out from vehicles in the other lane. Uses data and video feeds from cameras installed in taxis in Japan. TTV was lower in situations without crosswalks than those with crosswalks.

Had two classifications of pedestrians: unobstructed and obstructed. Obstructed was further broken down into 3 categories from behind a building, parked car, and moving car.

**Title: Behavioral Issues in Pedestrian Speed Choice and Street Crossing Behaviour: A Review<sup>9</sup>**

<https://doi.org/10.1080/01441640701365239>

Summary: Studies in the literature review showed that pedestrian speeds vary depending on crossing with the lowest being Zebra crossings, then pedestrian refuge, then pelican crossings, and the highest at random crossings. Delay was lowest at Zebra crossings, then random crossings, pedestrian refuges and the highest at pelican crossings. (Pelican crossings are primarily used in the UK, but are similar to HAWK signals. This suggests less risk is associated with slower speeds.

Pedestrian speed choice is also associated with the vehicle gaps. Different age groups accept different gaps.

**Title: Probabilistic Framework for Automated Analysis of Exposure to Road Collisions<sup>10</sup>**

<https://journals.sagepub.com/doi/10.3141/2083-11>

Summary: “Users are on a collision course when, “unless the speed and/or the direction of the road users changes, they will collide.” Scandinavian countries use safety hierarchy when defining road safety with collisions at the top.” “Among TCTs, the Swedish one is among the best known and is still being actively used for everyday safety assessments. It relies on the time to accident (TA), defined as “the time that is remaining from when the evasive action is taken until the collision would have occurred if the road users had continued with unchanged speeds and directions. Its value can be calculated based on the estimates of the distance to the potential point of collision and the speed when the evasive action is taken””

Provides formulas used for Collision Probability for two road users and three road users. Also defines methodology to define probability over a given time. Two major parameters: Conflicting Speed (CS): “speed of the road user taking evasive action, for whom the TA value is estimated, at the moment just before the start of the evasive action.” Time to Accident (TA): “time that remains to a crash from the moment that one of the road users starts an evasive action if they had continued with unchanged speeds and directions.”

Goes through formulas for how they calculated probability of collisions in the methodology section.

Video Processing: Uses an Image Sequence, Trajectory Database, and an Interaction Database. Algorithm relies on world coordinates through the estimation of the homography matrix. “The

motion pattern probabilities are computed by matching all trajectories over a given period through LCSS and can be updated continuously in a real-time application as traffic patterns change in time. When one computes the collision probability, the partial trajectories of each considered road user at each time are matched against the set of learned prototypes through LCSS.”

**Title: Estimating the severity of safety related behavior<sup>11</sup>**

<https://www.sciencedirect.com/science/article/pii/S0001457505001818>

Summary: Uses hierarchy to define road conflicts. A serious conflict is a situation where no one puts themselves in deliberately. Found that events at the highest severities in regards to pedestrian vehicle conflicts occur more frequently at signalized than non-signalized intersections. Those events of fairly high severities are the “predominant behaviour at the non-signalised intersection, while there are no crashes or serious conflicts. At the signalised intersection on the other hand, those interactions with fairly high severity do not seem to exist while crashes do.” Belief from the authors suggest that these “high severity” events are positive “because they are frequent and severe enough to produce efficient feedback to the involved road users, but not severe enough to result in crashes.”

Author also believes that a higher frequency of lower severities may not necessarily indicate high levels of safety since road users may not be prepared for events of higher severities and are likelier to end in a collision.

**Title: Methodology for Evaluating the Safety of Midblock Pedestrian Crossings<sup>12</sup>**

<https://journals.sagepub.com/doi/abs/10.3141/1828-09>

Summary: MBPCs are more dangerous crossings for pedestrians. Especially with darkness and alcohol. In Las Vegas, the MBPC spiked 150 feet from the nearest intersection.

**Title: Methodologies for Aggregating Traffic Conflict Indicators<sup>13</sup>**

<http://n.saunier.free.fr/saunier/stock/ismail11safety-index.pdf>

Summary: PET is a better indicator for representing severity of crossing events. Spatial proximity for pedestrians were chosen to be 10 m. A case study with a pedestrian scramble significantly reduced pedestrian vehicle conflicts. “The proposed safety measure is based on normalizing the summation of all severity indices by the maximum possible exposure.”

**Title: Illegal mid-block pedestrian crossings in China: gap acceptance, conflict and crossing path analysis<sup>14</sup>**

<https://www.tandfonline.com/doi/full/10.1080/17457300.2011.628751?scroll=top&needAccess=true>

Summary: From previous study, pedestrians crossing multiple lanes will not wait for all lanes to clear, rather a “rolling-gap”. Pedestrians classified as law abiding or opportunistic. Results showed most significant illegal occurrences occurred when the legal walking distance was 5 times the illegal one.

**Title: Development of a Conflict Technique for Pedestrian Crossings<sup>15</sup>**

<http://onlinepubs.trb.org/Onlinepubs/trr/1980/743/743-003.pdf>



Summary:

Conflict Definitions:

1. Slow or Weave for Walking Pedestrian: Occurs at when pedestrian is walking and occurs right angles
2. Slow or Weave for Running Pedestrian: Occurs at when pedestrian is running and occurs right angles
3. Pedestrian Walking or Running in the Roadway with the Flow of Traffic: Suggests need for sidewalks
4. Pedestrian Walking or Running in the Road Against the Flow of Traffic
5. Diagonal Pedestrian Crossing
6. Pedestrian in Center Lane
7. Outside Crosswalk
8. Right-Turning Conflicts
9. Left-Turning Conflicts
10. RTOR Conflicts
11. Signal Change
12. Pedestrian Violation
13. Vehicle Violation

Conflict Severity:

Hesitation, Backup Movement, Running Movement, Near Miss, PDO, Injury, Fatality

Routine, erratic, near miss, PDO, Injury, Fatality

Used manual observers combined with these categorizations.

**Title: Large-scale automated proactive road safety analysis using video data<sup>16</sup>**

<https://www.sciencedirect.com/science/article/pii/S0968090X15001485>

Summary: Analyzes differing roundabout locations in Canada. Multi-lane roundabouts are more likely to have larger amounts of vehicle conflicts depending on geometry.

Video Processing: "Points are evenly spaced in time with a consistent equivalent to the inverse of the frame rate of the video (typically 15–30 frames per second), i.e. a measurement is done for each frame. The object (road user) itself is represented by a group of characteristic features spread over the object and moving in unison."

Three errors occur: Parallax error, pixel resolution, and tracking errors. Used trajectory clustering for vehicle paths along roundabout.

**Title: Ch 7. Development of a Method for Determining Crash Modification Factors<sup>17</sup>**

[http://g92018.eos-intl.net/eLibSQL14\\_G92018\\_Documents/13-17.pdf](http://g92018.eos-intl.net/eLibSQL14_G92018_Documents/13-17.pdf)

Summary:

“Time to collision is probably a much better surrogate than distance to collision, as it accounts for reaction time, velocity, and braking distance. However, estimating time to collision requires trajectory data for both agents, which is typically difficult to obtain in case-study analysis.”

10<sup>th</sup> Street and Myrtle before and after implementation study. Average of one near miss / every 2 hours. Will need vehicle trajectories to be able to count TTC.

**Title: Large-Scale Automated Analysis of Vehicle Interactions and Collisions<sup>18</sup>**

<https://journals.sagepub.com/doi/abs/10.3141/2147-06>

Summary: Used extrapolation formula to dictate future position and crash probabilities. Only side and parallel interactions studied in detail.

Has formulas to describe how probability of collision is calculated. Probability calculations become much more complex once multiple road users are considered. Used proximity cut off of 1.7 m to represent an average vehicles width. Used a reference grid to calibrate the image.

**Title: Safety evaluation of right-turn smart channels using automated traffic conflict analysis<sup>19</sup>**

<https://www.sciencedirect.com/science/article/pii/S0001457511003204>

Summary: Before and after safety evaluation of “smart” right turn channel. Smart channels lead to improved sight distances and are at a 70 degree angle. Found that the 70 degree angle channels reduced conflicts significantly between vehicles, pedestrians were not mentioned.

Calibration and Processing: Used 6 extrinsic parameters and 2 intrinsic parameters to calibrate the view. Used an LCSS matching algorithm to generate the conflict events.

**Title: Traffic conflict standards for intersections<sup>20</sup>**

<https://www.tandfonline.com/doi/pdf/10.1080/03081069908717634?needAccess=true>

Summary: Traffic conflicts at intersections between cars in both signalized and unsignalized intersections. Developed an intersection conflict index using average hourly conflict rate and average severe conflict rate as well as PEV which is the square root of the product of the hourly entering volumes in thousands

**Title: Automated Analysis of Road Safety with Video Data<sup>21</sup>**

<https://journals.sagepub.com/doi/10.3141/2019-08>

Summary: Various methods used to get vehicle conflicts. Extrapolating vehicle trajectories is the one that is the most common. “Two approaches involve learning for traffic conflict detection: (a) supervised learning for interaction classification and (b) unsupervised learning of vehicle dynamics and movements for prediction. Uses feature based tracking and The algorithm relies on world coordinates through the estimation of the homography matrix.”

“The clustered observations are therefore sequences of four-dimensional vectors, composed of the vehicle coordinates (x, y) and velocity (sx, sy) at each time step. The estimated size of the vehicles should also be useful, but it did not improve the results in the experiments. Since the

trajectories obtained through the vehicle tracking algorithm are noisy, they are smoothed by using a moving average filter”

“The detection process proceeds as follows:

1. Vehicles are tracked;
2. If two vehicles are close enough (threshold on their distance) and nearing each other (their distance decreases), they are considered to be in interaction;
3. Each interacting vehicle trajectory is assigned to an HMM, say, A and B; and
4. If the two HMMs A and B of both interacting trajectories were both memorized as conflicting both memorized as conflicting, a traffic conflict between these two vehicles is detected.”

**Title: Traffic monitoring and accident detection at intersections<sup>22</sup>**

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=880968>

Summary: Discusses different vehicle tracking methodologies, uses a spatio-temporal Markov random field (MRF) for traffic images at intersections. Used a hidden Markov model to recognize events such as rear end bumper crashes, passing, and jamming.

**Title: Road User Collision Prediction Using Motion Patterns Applied to Surrogate Safety Analysis<sup>23</sup>**

<http://n.saunier.free.fr/saunier/stock/st-aubin14motion-patterns.pdf>

Summary: Uses probability model and automatic processing to determine if a road user will be in a certain location at a certain time. Small step size is needed for precise predictions. Gives equations to calculate each. Demonstrates that using these predicted patterns it remains possible to be a better surrogate measure of safety than maintaining constant velocity.

**Title: PEDESTRIAN-VEHICLE CONFLICT ANALYSIS AT SIGNALIZED INTERSECTIONS USING MICRO-SIMULATION<sup>24</sup>**

<http://vti.diva-portal.org/smash/get/diva2:926089/FULLTEXT01.pdf>

Summary: Uses Vissim to try and replication ped-vehicle conflicts. The study mentions the Surrogate Safety Assessment Model designed by the FHWA. Two different types of ped-vehicle conflicts. Vehicle yield to pedestrians, and pedestrians yield to vehicles. Vehicle-yield-ped conflict more dangerous. Vissim trajectory data was transferred to the SSAM to generate TTC and PET. Simulated crashes were slightly lower in some cases since peds have to obey signals 100% of the time in Vissim. Max TTC found to be 2.7 seconds and max PET was found to be 8 s to generate the best results.

**Title: Performance Evaluation and Correction Functions for Automated Pedestrian and Bicycle Counting Technologies<sup>25</sup>**

<https://ascelibrary.org/doi/10.1061/%28ASCE%29TE.1943-5436.0000828>

Summary: Evaluated different type of automatic devices to count peds and bikes.

**Title: Prediction of drivers and pedestrians' behaviors at signalized mid-block Danish offset crosswalks using Bayesian networks<sup>26</sup>**

<https://www.sciencedirect.com/science/article/pii/S0022437518306595?via%3Dihub>

Summary: Estimates driver and pedestrian compliance rates at push-button intersections uses Bayesian Networks. Uses observational data from Denmark and Las Vegas. How likely are pedestrians to push the button given traffic conditions and how likely are cars to yield with flashing lights? Gave probabilities for differing flashing treatments, situations, and locations.

**Title: A methodology for precise camera calibration for data collection applications in urban traffic scenes<sup>27</sup>**

<https://www.nrcresearchpress.com/doi/pdf/10.1139/cjce-2011-0456>

Summary: Repainting makes it difficult for points localization. Practical motivation section goes into detail about the issues. Existing techniques rely on parallel vehicle tracks as well which may be an inaccurate assumption. Existing camera calibration methods include using existing geometry, self-calibration with epipolar, and active vision calibration. Goes through their calibration model and has found more success when comparing to previous attempts when relating to real world distance. The intrinsic camera parameters optimized under calibration are focal length, skew angle, and radial lens distortion. The extrinsic parameters are the translation and rotation (six parameters) of the camera coordinate system from the world coordinate system.

**Title: Analysis of car-to-bicycle approach patterns for developing active safety devices<sup>28</sup>**

<https://www.tandfonline.com/doi/full/10.1080/15389588.2015.1087641>

Yasuhiro Matsui, Shoko Oikawa & Masahito Hitosugi (2016) Analysis of car-to-bicycle approach patterns for developing active safety devices, *Traffic Injury Prevention*, 17:4, 434-439, DOI: [10.1080/15389588.2015.1087641](https://doi.org/10.1080/15389588.2015.1087641)

Summary: Same procedure as pedestrian study. Obstructed cyclists were more likely to have traffic conflicts. Very similar to other pedestrian study.

**Title: The extreme value theory approach to safety estimation<sup>29</sup>**

<https://www.sciencedirect.com/science/article/pii/S0001457506000236>

Summary: Focused on straight right angle collisions and left turning collisions. Previous study concluded PET was the most promising indicator for safety implication. Used a method where risk was measured “with a time interval during which more than one pair of vehicles have an opportunity to collide.” Observance time was 8 hours and 18 selected sites. Listed the steps to apply their method:

1. Crash proximity measured corresponding with the studied type of crash must be defined.
2. A valid crash proximity measure must be observable and possess a continuous characteristic that can represent crash-free operations as well as characterize a collision.
3. A definitive boundary between crash and non-crash must exist.

4. The risk estimation method should include a bias-variance trade-off, a choice of r-value, and identification of non-stationarity and associated covariates.

Overestimated crashes in intersections that had queuing into the intersection.

*\*Vehicle speed is also an important factor*

## References

- [1] Ismail, Karim, Tarek Sayed, Nicolas Saunier, and Clark Lim. "Automated Analysis of Pedestrian–Vehicle Conflicts Using Video Data." *Transportation Research Record* 2140, no. 1 (January 2009): 44–54. doi:10.3141/2140-05.
- [2] Malkhamah, Siti, Miles Tight, and Frank Montgomery. "The Development of an Automatic Method of Safety Monitoring at Pelican Crossings." *Accident Analysis & Prevention* 37, no. 5 (2005): 938–46. <https://doi.org/10.1016/j.aap.2005.04.012>.
- [3] Sayed, Tarek, Mohamed H. Zaki, and Jarvis Autey. "Automated Safety Diagnosis of Vehicle–Bicycle Interactions Using Computer Vision Analysis." *Safety Science* 59 (2013): 163–72. <https://doi.org/10.1016/j.ssci.2013.05.009>.
- [4] Ismail, Karim, Tarek Sayed, Mohamed H. Zaki, and Fahad Alrukaibi. "Automated Detection of Spatial Traffic Violations through Use of Video Sensors." *Transportation Research Record* 2241, no. 1 (January 2011): 87–98. doi:10.3141/2241-10.
- [5] Laureshyn, A., & Ardö, H. (2006). Automated video analysis as a tool for road user behavior. In R. Risser (Ed.), [Host publication title missing] *Proceedings of ITS World Congress, London, 8-12 October 2006*.
- [6] Jiang, Xiaobei, Wuhong Wang, Klaus Bengler, and Weiwei Guo. "Analyses of Pedestrian Behavior on Mid-Block Unsignalized Crosswalk Comparing Chinese and German Cases." *Advances in Mechanical Engineering*, (November 2015). doi:10.1177/1687814015610468.
- [7] Schneider, Robert J, Rhonda M Ryznar, and Asad J Khattak. "An Accident Waiting to Happen: a Spatial Approach to Proactive Pedestrian Planning." *Accident Analysis & Prevention* 36, no. 2 (2004): 193–211. [https://doi.org/10.1016/s0001-4575\(02\)00149-5](https://doi.org/10.1016/s0001-4575(02)00149-5).
- [8] Matsui, Yasuhiro, Masahito Hitosugi, Tsutomu Doi, Shoko Oikawa, Kunio Takahashi, and Kenichi Ando. "Features of Pedestrian Behavior in Car-to-Pedestrian Contact Situations in Near-Miss Incidents in Japan." *Traffic Injury Prevention* 14, no. sup1 (2013). <https://doi.org/10.1080/15389588.2013.796372>.
- [9] Ishaque, Muhammad Moazzam, and Robert B. Noland. "Behavioural Issues in Pedestrian Speed Choice and Street Crossing Behaviour: A Review." *Transport Reviews* 28, no. 1 (2008): 61–85. <https://doi.org/10.1080/01441640701365239>.
- [10] Saunier, Nicolas, and Tarek Sayed. "Probabilistic Framework for Automated Analysis of Exposure to Road Collisions." *Transportation Research Record* 2083, no. 1 (January 2008): 96–104. doi:10.3141/2083-11.
- [11] Svensson, Åse, and Christer Hydén. "Estimating the Severity of Safety Related Behaviour." *Accident Analysis & Prevention* 38, no. 2 (2006): 379–85. <https://doi.org/10.1016/j.aap.2005.10.009>.
- [12] Cui, Zhenzhong, and Shashi S. Nambisan. "Methodology for Evaluating the Safety of Midblock Pedestrian Crossings." *Transportation Research Record* 1828, no. 1 (January 2003): 75–82. doi:10.3141/1828-09.

- [13] Ismail, Karim, Tarek Sayed, and Nicolas Saunier. "Methodologies for Aggregating Indicators of Traffic Conflict." *Transportation Research Record: Journal of the Transportation Research Board* 2237, no. 1 (2011): 10–19. <https://doi.org/10.3141/2237-02>.
- [14] Cherry, Christopher, Brian Donlon, Xuedong Yan, Samuel Elliott Moore, and Jian Xiong. "Illegal Mid-Block Pedestrian Crossings in China: Gap Acceptance, Conflict and Crossing Path Analysis." *International Journal of Injury Control and Safety Promotion* 19, no. 4 (2012): 320–30. <https://doi.org/10.1080/17457300.2011.628751>.
- [15] Cynecki, Michael. "Development of Conflicts Analysis Technique for Pedestrian Crossings". In *Transportation Research Record* 743, TRB, National Research Council, Washington, D.C., 1980, pp. 12–20.
- [16] St-Aubin, Paul, Nicolas Saunier, and Luis Miranda-Moreno. "Large-Scale Automated Proactive Road Safety Analysis Using Video Data." *Transportation Research Part C: Emerging Technologies* 58 (2015): 363–79. <https://doi.org/10.1016/j.trc.2015.04.007>.
- [17] Watkins, Kari, Michael Rodgers, Randel Guensler, and Yanzhi Xu. "BICYCLE AND PEDESTRIAN SAFETY IN THE HIGHWAY SAFETY MANUAL." *GDOT*. July 2016.
- [18] Saunier, Nicolas, Tarek Sayed, and Karim Ismail. "Large-Scale Automated Analysis of Vehicle Interactions and Collisions." *Transportation Research Record* 2147, no. 1 (January 2010): 42–50. doi:10.3141/2147-06.
- [19] Autey, Jarvis, Tarek Sayed, and Mohamed H. Zaki. "Safety Evaluation of Right-Turn Smart Channels Using Automated Traffic Conflict Analysis." *Accident Analysis & Prevention* 45 (2012): 120–30. <https://doi.org/10.1016/j.aap.2011.11.015>.
- [20] Sayed, Tarek, and Sany Zein. "Traffic Conflict Standards for Intersections." *Transportation Planning and Technology* 22, no. 4 (1999): 309–23. <https://doi.org/10.1080/03081069908717634>.
- [21] Saunier, Nicolas, and Tarek Sayed. "Automated Analysis of Road Safety with Video Data." *Transportation Research Record: Journal of the Transportation Research Board* 2019, no. 1 (2007): 57–64. <https://doi.org/10.3141/2019-08>.
- [22] Kamijo, S., Y. Matsushita, K. Ikeuchi, and M. Sakauchi. "Traffic Monitoring and Accident Detection at Intersections." *IEEE Transactions on Intelligent Transportation Systems* 1, no. 2 (2000): 108–18. <https://doi.org/10.1109/6979.880968>.
- [23] St-Aubin, Paul. "Road User Collision Prediction Using Motion Patterns Applied to Surrogate Safety Analysis." (2014).
- [24] Wu, Jiawei, Essam Radwan and Hatem Abou-Senna. "Pedestrian-vehicle conflict analysis at signalized intersections using micro-simulation." (2016).
- [25] Proulx, Frank R., Robert J. Schneider, and Luis F. Miranda-Moreno. "Performance Evaluation and Correction Functions for Automated Pedestrian and Bicycle Counting Technologies." *Journal of Transportation Engineering* 142, no. 3 (2016): 04016002. [https://doi.org/10.1061/\(asce\)te.1943-5436.0000828](https://doi.org/10.1061/(asce)te.1943-5436.0000828).



- [26] Kutela, Boniphace, and Hualiang Teng. "Prediction of Drivers and Pedestrians Behaviors at Signalized Mid-Block Danish Offset Crosswalks Using Bayesian Networks." *Journal of Safety Research*69 (2019): 75–83. <https://doi.org/10.1016/j.jsr.2019.02.008>.
- [27] Ismail, Karim, Tarek Sayed, and Nicolas Saunier. "A Methodology for Precise Camera Calibration for Data Collection Applications in Urban Traffic Scenes." *Canadian Journal of Civil Engineering*40, no. 1 (2013): 57–67. <https://doi.org/10.1139/cjce-2011-0456>.
- [28] Matsui, Yasuhiro, Shoko Oikawa, and Masahito Hitosugi. "Analysis of Car-to-Bicycle Approach Patterns for Developing Active Safety Devices." *Traffic Injury Prevention*17, no. 4 (2015): 434–39. <https://doi.org/10.1080/15389588.2015.1087641>.
- [29] Songchitruksa, Praprut, and Andrew P. Tarko. "The Extreme Value Theory Approach to Safety Estimation." *Accident Analysis & Prevention*38, no. 4 (2006): 811–22. <https://doi.org/10.1016/j.aap.2006.02.003>.

## Appendix 5 – Technical Document for Conflict Analysis

### General Overview:

The output of the multimon and reportgen provides the frame by frame coordinates of each unique object which the tracker identifies. The tracker outputs these bounding box coordinates in terms of (x,y) position as well as width and height. For each unique tracked object, the tracker also provides the object type (cyclist, pedestrian, vehicle), frame id, and unique object id. All these outputs are used to identify conflicts.

### Workflow:

The code for the conflict analysis pulls directly from the outputs of reportgen. After using pedmondetect to process a video with a specific Site ID, running reportgen provides filters out some false positives, such as a traffic light being labeled a pedestrian. The Conflict Analysis code works directly from the reportgen outputs and file structure. The user only needs to select the report directory and choose the cutoff points for the analysis. The *Conflict Analysis Tool Manual* in Appendix 1 shows the steps in detail.

### Point Filtering

After inputting reportgen's output csv files, the code uses a time based filter to see whether or not trajectories intersect. Essentially, each position for pedestrians and vehicles are points. Since the location are points, it is not a trivial problem to see whether or not the objects intersect with one another. Line segments could be created for all unique vehicles and pedestrians and intersected with the opposing class. This exercise, however, would provide thousands of potential "conflicts" without knowing whether or not they actually occur within a time window.

To address the importance of time, a sliding time window is applied. Currently the window used is a 10 second window with timestep of 1 second. While using a 1 second timestep takes longer to process than using a larger timestep, it ensures that all conflicts are accounted for. Based on the existing literature, an interaction is considered a slight conflict between pedestrians and vehicles when the post-encroachment time (PET), the time difference when two objects are at a certain point, is between 3-5 s. A conflict exists when the PET is between 1-3 s. A severe conflict exists when the PET is lower than 1. The current applied 10 second window can therefore likely be smaller, but should still be slightly greater than 5 seconds.

### Intersection Assumptions

For a time based filtering system, an intersection between two unique objects is believed to occur in the:

#### Horizontal Direction when:

A unique vehicle ID has at least two points within the minimum and maximum x trajectory with the given time window. AND a unique vehicle ID which has a y trajectory below the pedestrian y minimum AND which has a y trajectory above the maximum y during that time period.

The code first filters through and checks to see which unique vehicle IDs are within the time frame for ONE unique pedestrian and then checks to see whether the same unique vehicle ID has a point both above and below the pedestrian trajectories.

Figure 94 below depicts a sample horizontal conflict the code will detect.

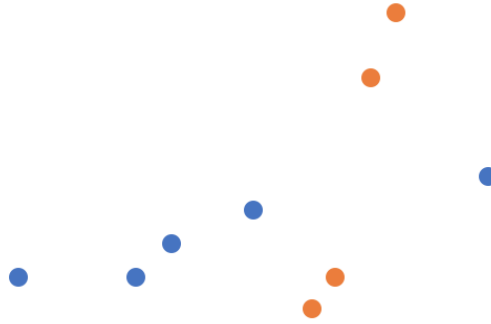


Figure 94 - Sample horizontal conflict with pedestrian trajectory in blue and vehicle trajectory in orange

The blue dots represent a possible pedestrian track, and the orange dots represent a possible vehicle track. While no points overlap, the vehicle track has points within the x-axis bound of the pedestrian, and also has points both above and below the min and max y values. The interpolation section will discuss how the post-encroachment time is calculated for conflicts.

**Vertical Direction** when:

A unique vehicle ID has at least two points within the minimum and maximum y trajectory with the given time window **AND** a unique vehicle ID which has a x trajectory to the left of the pedestrian x minimum **AND** which has a x trajectory to the right of the maximum x during that time period. These type of conflicts usually occur when a pedestrian is walking along the roadway and crosses a driveway or intersection. Figure 95 below depicts a sample vertical conflict.



Figure 95 - Sample vertical conflict with pedestrian trajectory in blue and vehicle trajectory in orange

Like with the sample horizontal conflict, the blue dots represent a sample pedestrian track, while the orange dots represent a sample vehicle track. Here, the vehicle track is outside the horizontal, and within the vertical bound bounds of the pedestrian track.

Using a relatively large window, short time-step, and categorizing these conflict types finds all possible conflicts from the reportgen outputs.

## Step Through:

The code first filters through and checks to see which unique vehicle IDs are within the time frame for **one** unique pedestrian, and then checks to see whether the same unique vehicle ID has an additional point that falls within either a horizontal or vertical conflict category.

If a conflict does exist, the unique pedestrian ID, unique vehicle ID, x and y trajectories, specific time frames, and conflict category throughout the entire time window are put into a data frame called **conflicts**.

A more readable data frame called **conflict\_table** exists where the time window, pedestrian and vehicle ID which conflict, and intersection category. Once the code filters conflicts for the entire reportgen output, interpolation occurs for those conflicts. For a large dataset, the computational time is not insignificant. For a dataset of 4 cameras of 2.5 days, the processing time takes about 18 hours.

## Interpolation:

Now that all conflicts are accounted for, interpolation must happen to know *when* the paths intersect to calculate PET. While a Python package has a built in function which calculates the intersection of two linestrings, this package does not output the points which cause the intersection. While it outputs the points of the intersect, the points which are closest geometrically to the intersection point may or may not reflect the true intersection.

To capture the accurate interpolated intersection, multiple nested loops are used. For each conflict in the **conflict\_table** dataframe, the pedestrian and vehicle trajectories are obtained for the given time window.

First, the pedestrian and vehicle trajectories are compared to see whether an pedestrian and vehicle intersect at the exact point. If both trajectories share the same x,y coordinates, then that is the intersection point and the PET time can be calculated by finding the time difference of the two points.

For a vast majority of conflicts, however, no exact overlap between pedestrian and vehicle tracks occurs. To see where the pedestrian and vehicle lines intersect, each piecewise segment of the entire pedestrian track is checked against **all** piecewise vehicle segments of the entire vehicle track. If no intersection exists for that specific pedestrian segment, then the next pedestrian piecewise segment is checked against **all** vehicle segments. This loop continues until an intersection occurs between a pedestrian and vehicle segment\*.

*\*Even for a large dataset, this process occurs relatively quickly. For the 2.5 day, 4 camera dataset, the intersection checking process only took about 30 minutes.*

Once an intersection occurs, the distance between the first point on both the pedestrian and vehicle lines to the intersection point is calculated. The time it takes to travel from the start to end of the respective pedestrian and vehicle segment is also calculated. The time to the intersection point is then calculated assuming a constant speed between the two points.

*Note: Speed between the two points is likely not constant, but currently it is the best way to calculate time differentials. The points are likely close enough in physical distance that the speed should not drastically change over that segment.*

The PET is then calculated by subtracting the pedestrian time from the vehicle time:

$$\text{PET} = \text{Vehicle Intersection Time} - \text{Pedestrian Intersection Time}$$

If the PET time is negative, that means the pedestrian passed behind the vehicle which is a less severe type of conflict. If the PET time is positive, that means the vehicle passed behind the pedestrian which is comparatively more severe type of conflict.