Bigdata Analytics and Artificial Intelligence for Smart Intersections

Final Report for FDOT Project BDV31-977-116

Submitted to

Raj Ponnaluri, Ph.D., P.E., PTOE

Florida Department of Transportation

Submitted by

Tania Banerjee, Ke Chen, Xiaohui Huang, Pan He, Patrick Emami, Aotian Wu, Lily Elefteriadou, Anand Rangarajan, Siva Srinivasan, and Sanjay Ranka

University of Florida Transportation Institute

May 2022

Disclaimer

Technical Report Documentation Page

1. Report No.	**2. Government Accession No.**	**3. Recipient's Catalog No.**	
4. Title and Subtitle **Bigdata Analytics and Artificial Intelligence for Smart Intersections**		**5. Report Date** June 2022	
		6. Performing Organization Code	
7. Author(s) Tania Banerjee, Ke Chen, Xiaohui Huang, Pan He, Patrick Emami, Aotian Wu, Lily Elefteriadou, Anand Rangarajan, Siva Srinivasan, and Sanjay Ranka		**8. Performing Organization Report No.**	
9. Performing Organization Name and Address UF Transportation Institute College of Engineering University of Florida Gainesville, FL 32611		**10. Work Unit No. (TRAIS)**	
		11. Contract or Grant No. BDV31-977-116	
12. Sponsoring Agency Name and Address Florida Department of Transportation 605 Suwannee Street, MS 30 Tallahassee, FL 32399		**13. Type of Report and Period Covered Final Report** Final Report 07/03/2019 – 06/30/2022	
		14. Sponsoring Agency Code	
15. Supplementary Notes			

16. Abstract

This project aims to develop a multi-sensor system for vehicle and pedestrian traffic analysis at traffic intersections. The techniques developed as part of this project are used to process data streams from video-camera and LIDAR systems installed at traffic intersections (along with the loop detector data captured by advanced traffic controllers). We use state-of-the-art computer vision and machine learning to perform vehicle tracking (localization, tracking, turn estimation) and pedestrian monitoring from data streams obtained using standard fisheye cameras and LIDARs mounted at intersections and loop detectors beneath the streets. This tracking information is then used to derive motion patterns or trajectories. The trajectories can be utilized for traffic conflict detection and safety analysis and to improve signal timing plans for increased traffic throughput. LIDAR data is beneficial for counting the pedestrians and bicyclists and tracking the patterns of movement they follow while crossing the intersection especially when lighting conditions are poor.

17. Key Word Video Analytics, Transportation Applications, Machine Learning, Safety Analysis		**18. Distribution Statement** No restrictions.	
19. Security Classif. (this report) Unclassified.	**20. Security Classif. (this page)** Unclassified.	**21. No. of Pages** 165	**22. Price**

Acknowledgments

The authors would like to thank the Florida Department of Transportation (FDOT) for providing funding support

Executive Summary

Vehicle loop detectors that have traditionally been deployed at intersections to detect the passage of vehicles have high deployment and maintenance costs; and are not always useful for observing the movements of pedestrians and scooters. The use of other modalities, such as video and LIDAR, has great potential to improve accuracy and timeliness in the detection of vehicles, pedestrians, bicyclists, etc.

This report demonstrates the use of state-of-the-art edge-based video-stream processing, infrared cameras, and LIDAR at intersections to convert video data into space-time trajectories of individual vehicles and pedestrians that are transmitted and synthesized on a cloud-based system. Using standard and fisheye cameras mounted at intersections (and integrating loop detectors placed beneath streets), we used state-of-the-art computer vision and machine learning to perform vehicle tracking (localization, tracking, turn estimation) and pedestrian monitoring at intersections. This report summarizes our work to develop and test (as needed) the following:

- Video and LIDAR processing techniques and software that can work with a variety of vehicular traffic, pedestrian traffic, changing conditions, and multiple video cameras per intersection.
- Nighttime camera processing techniques and software suitable for effectively observing and analyzing nighttime traffic behavior.
- Algorithms and software for determination of near-misses, anomalous vehicle trajectories, and incidents.
- Software for near-misses and other anomalous behavior at the intersection.
- Algorithms for signal optimization that use information from all the video and LIDAR sensors along with information collected from connected and autonomous vehicles.

Using data collected from a number of traffic intersections, we demonstrate our impact on understanding traffic behavior at an intersection that can lead to:

- Improved pedestrian and bicyclists safety by examining the conflict points of the vehicle and pedestrian trajectories and anomalous behavior, based on time of day and day of the week.
- More accurate demand profiles, which can lead to more effective signal timing and a better understanding of the relationships between existing (loop detector) and new technologies (video).
- Better incident and bottleneck management using real-time information of incidents and other events.

Table of Contents

List of Figures

List of Tables

Chapter 1 – Develop Algorithms and Software Video Processing-based Traffic Analysis for Single and Multiple Fisheye Cameras

1.1. Introduction

The Florida Department of Transportation is interested in using a variety of data sources (road sensors, roadside video, etc.) to comprehensively analyze the flow of vehicles and commodities across the state of Florida. The advent of nominally priced video-based systems, open source tools for video processing and deep learning, and the availability of low cost graphics processing units (GPUs) has opened the door for their use in real-time transportation decision systems. While video-based systems for intersection traffic measurement can perform multiple object detection and tracking, their use for more complex tasks such as anomaly detection and near-misses is limited. As a part of the first deliverable, we focused on building a platform that allows one to collect sufficient samples and visual cues corresponding to near-misses, with the goal of detecting and even anticipating dangerous scenarios in real-time so that appropriate preventive steps can be undertaken.

In order to derive intersection scenes with wider angles, omnidirectional fisheye cameras are widely installed and used for street video surveillance (also known as closed-circuit television, or CCTV). It is nontrivial to directly apply learning-based methods to detect near-misses or other traffic anomalies in fisheye videos because they suffer from two types of distortions: fisheye lens distortion and perspective distortion. Due to both these distortions, road users (pedestrians, cars, etc.) can appear to be very close to each other in image space and to the human eye while remaining far apart in the physical world. Figure 1-1 illustrates one real fisheye traffic scene and some false near-miss cases that can easily mislead.

In this report, we focus on near-miss problems from large-scale intersection videos collected from fisheye cameras. The goal was to temporally and spatially localize and recognize near-miss cases from video. To the best of our knowledge, currently, there is no literature on near-miss detection in fisheye traffic video. The motivations of resolving distortion instead of using original fisheye videos are to compute accurate distance among objects for spatial near-miss detection and to compute accurate speed information for temporal near-miss detection. Therefore, we aimed at detecting and tracking road objects in fisheye video and approximately projecting their locations in the real-world on an overhead satellite map of the intersection. The corrected trajectories and speed information of road objects can be utilized to detect near-misses in spatial and temporal domains, respectively. We specify five categories of objects of interest: pedestrians, motorbikes, cars, buses, and trucks. The overhead satellite maps of intersections are cropped from Google Earth®. The main steps of our detection framework (Figure 1-2) can be summarized as follows:

1. Fisheye-to-Cartesian Mapping Using Calibration and Thin-Plate Spline: We first apply camera calibration methods to a fisheye background image (with no road objects) to make an initial correction. We take the calibrated image as the target image and an overhead satellite map as the reference image and select corresponding landmark points in both images for mapping. Given these landmark points, we adopt the thin-plate spline (TPS) (Bookstein, 1989; Chui and Rangarajan, 2003) as the basis function for coordinate mappings from the reference to the target and store the point-to-point outputs.
2. Object Detection and Multiple Object Tracking: We train an object detector using deep learning techniques and design a vehicle re-identification model with deep cosine metric learning to handle

occlusion problems. We integrate these two models into our multiple object tracking pipeline. Given fisheye videos as the input, the framework supports real-time object detection and multiple object tracking.

3. Trajectory and Speed Computation: Using the point-to-point TPS mappings, we correct and scale road object trajectories and speed information from the perspective of the overhead satellite map with learned deep features. As the complexity of coordinates transfer is O(1), it allows us to process data both online and offline.
4. Spatial and Temporal Near-Miss Detection: We define two scenarios for near-misses in videos: (1) spatial scenario: close proximity of road objects in image space and (2) temporal scenario: a dramatic speed decrease to avoid near-misses (a sudden break). We use the distance-based and speed-based measures to compute the near-miss probabilities of road objects and aggregate scores via averaging as the final output.

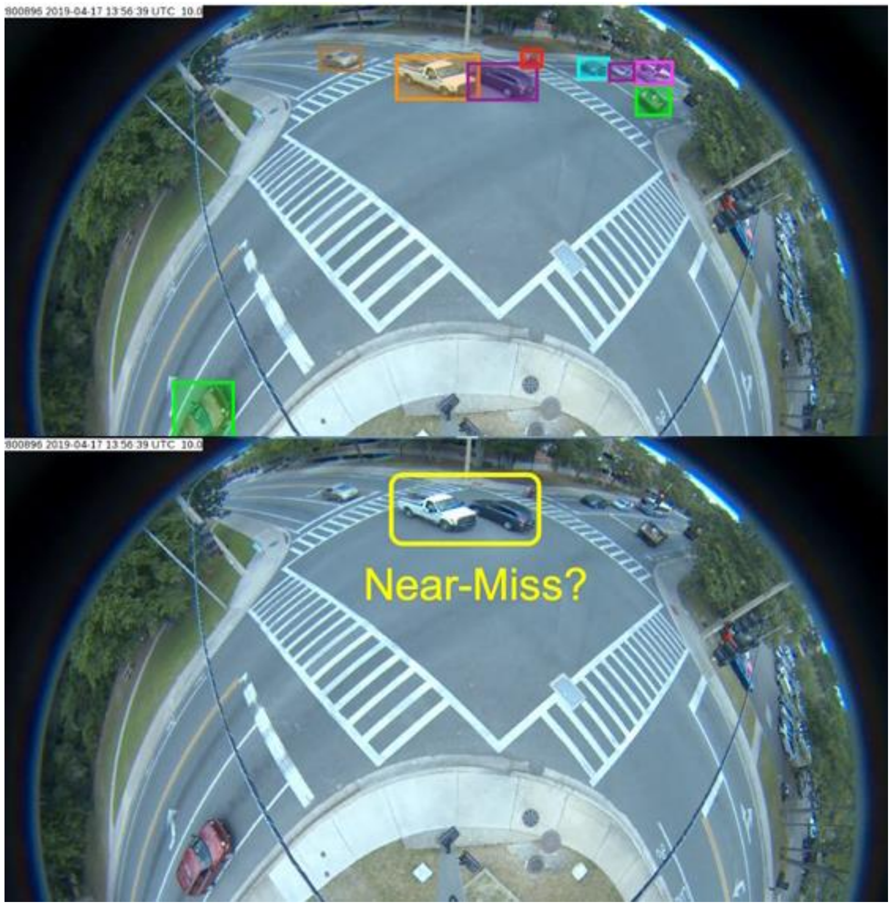

800896 2019-04-17 13:56:39 UTC
800896 2019-04-17 13:56:39 UTC
Near-Miss?

Figure 1-1. Top: Fisheye camera image of an intersection with bounding boxes on detected objects. Bottom: Bounding boxes indicate a near-miss, but this is a false identification. The two vehicles are driving through the intersection in opposite direction.

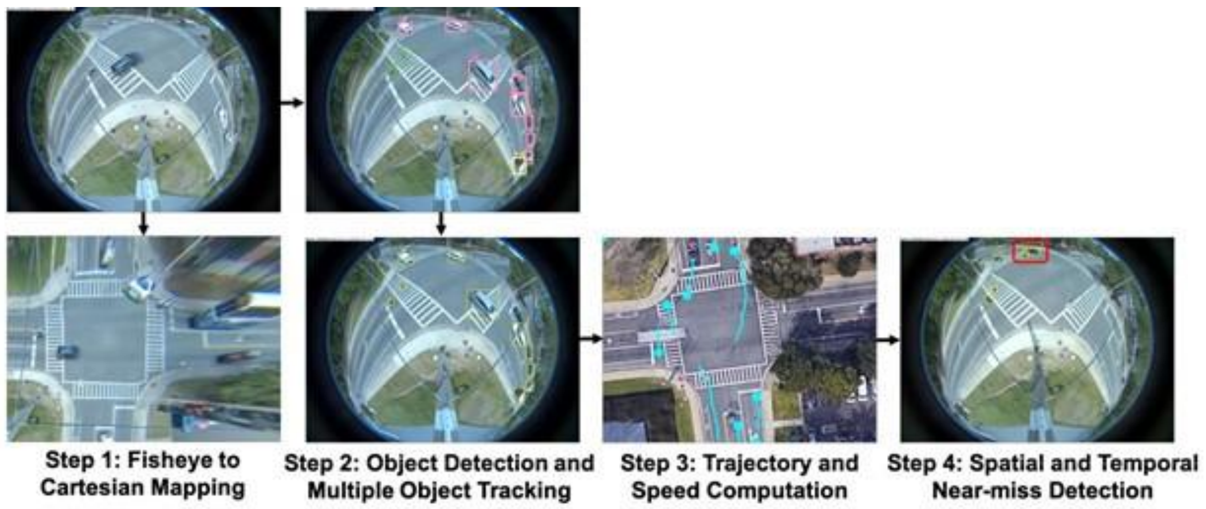

Step 1: Fisheye to Cartesian Mapping
Step 2: Object Detection and Multiple Object Tracking
Step 3: Trajectory and Speed Computation
Step 4: Spatial and Temporal Near-miss Detection

Figure 1-2. The pipeline overview of the proposed detection framework.

We developed a novel method that combines distance measures and temporal motion to detect near-misses in fisheye traffic video. Our calibration method is combined with a spline-based mapping method that maps fisheye video features to an area map to correct fisheye lens distortion and camera perspective distortion. We developed a unified approach that performs real-time object recognition, multiple object tracking, and near-miss detection in fisheye video. In this report we show a promising pipeline that can be used to customized to several applications that interpret fisheye video to yield accident anticipation, anomaly detection, and trajectory prediction.

We have obtained accurate and real-time object detection and multiple object tracking results for long-duration video. The object detector has good performance to localize and classify road objects even for tiny pedestrians. With cosine metric learning, the tracker generates more consistent and robust tracks and trajectories. With aid of calibration and TPS mappings, the coordinates and speed information of each object has been corrected and scaled to a large extent. Compared to methods not based on mapping, the experimental results demonstrated that our methods perform better in filtering out false near-miss cases.

1.2. Fisheye Video Data

To our best knowledge, we know of no comprehensive traffic video dataset consisting of omnidirectional fisheye videos for near-misses or other traffic analysis. We collected large-scale fisheye traffic video from four omnidirectional cameras at three intersections. Figure 1-3 shows the fisheye data sample from these cameras, showing the diversity of the intersection and lighting conditions. For the large intersection, there exist two cameras placed at diagonal corners of the intersection, and each has a perspective of more than half of the intersection. In this paper, we do not discuss analysis for the large intersection because removing fisheye distortion and stitching components are involved. We experimented on single-camera intersection videos and present qualitative and quantitative evaluation for weekly real traffic videos. We collected 8 hours of video on a daily basis for each intersection: 2 hours each for the morning, noon, afternoon, and evening. Total duration of all fisheye videos used in the experiments was more than 100 hours. Our fisheye intersection videos are more challenging than videos in other datasets collected by surveillance camera for reasons such as fisheye distortion, multiple object types (pedestrians and vehicles), and diverse lighting conditions. For generating ground truth for object detection, tracking, and near-miss detection, we annotated the spatial location (bounding boxes) and temporal location (frames) for each object and near-miss as well as their vehicle class in videos for each intersections.

Figure 1-3. Examples of fisheye video collected in different locations (4 cameras) under different lighting conditions

1.3. Qualitative Performance

1.3.1. Fisheye to Cartesian Mapping

We present qualitative results for calibration+TPS pipeline in Figure 1-4. It demonstrated that fisheye distortion and perspective distortion are well removed from original fisheye images. We present qualitative results in terms of performance for object detection, multiple object tracking, and superpixel segmentation in the Figure 1-5. The results demonstrated that the deep learning detector performed well in detecting road objects and assigning them to the correct class, even for tiny objects, e.g., pedestrians and motorbike. With the aid of deep cosine metric learning, the tracker generated more consistent and stable tracks. The superpixel segmentation also performed well in clustering and outputting compact contours of objects, which can be utilized for specific tasks with a desire of accurate object mask.

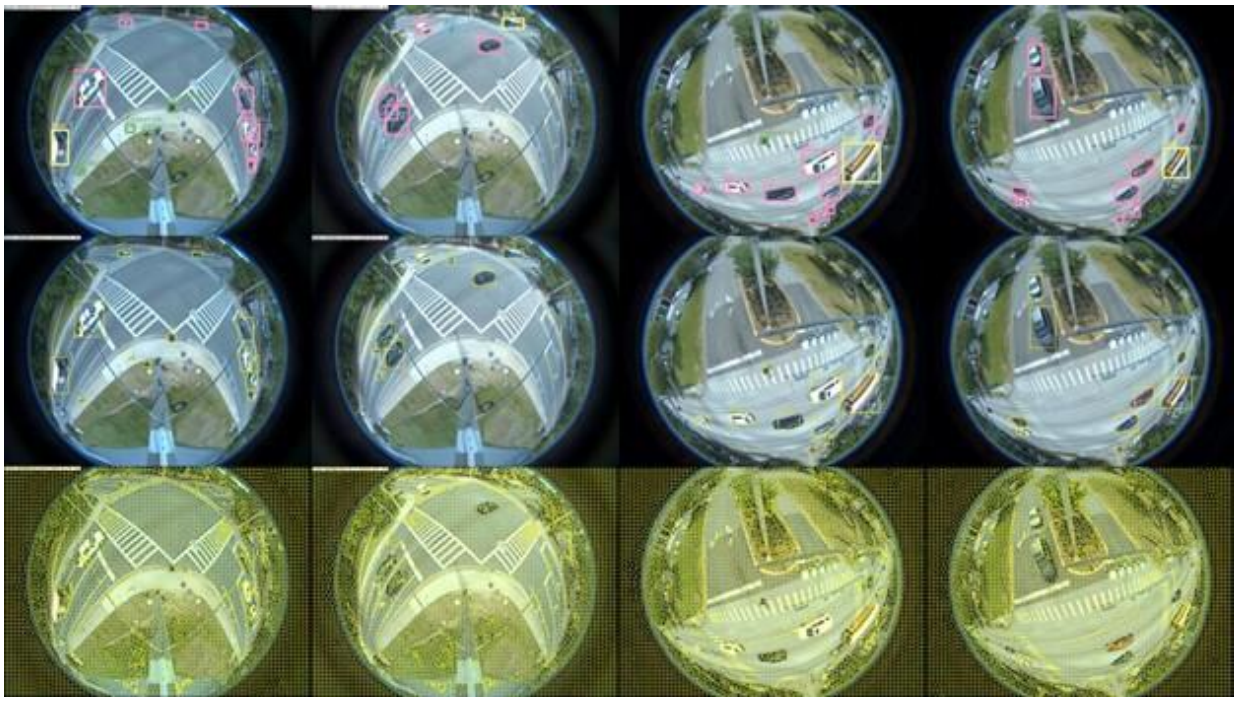


Figure 1-4. Calibration and thin-plate spline (TPS) mapping pipeline. Left: original fisheye image. Middle: mapping result. Right: reference satellite map

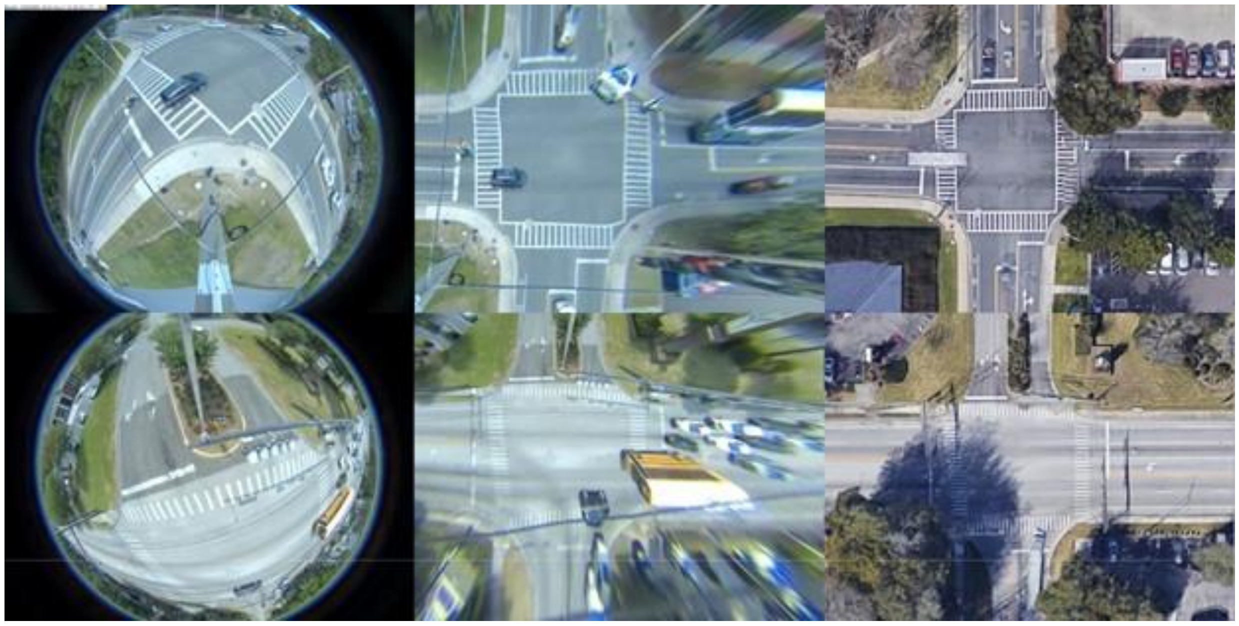


Figure 1-5. Top: object detection. Middle: multiple object tracking. Bottom: superpixel segmentation

1.3.2 Trajectory and Near-miss Detection

We presented the trajectories of road objects projected on the satellite map along with referenced tracking frames in Figure 1-6. These trajectories are formed by correct coordinates using a mapping points-set generated by calibration, perspective correction and TPS mapping pipeline. These trajectory maps give a cleaner traffic pattern for the intersection than that from the perspective of the original fisheye camera. Finally, we present some samples of near-misses we detected at different intersections in Figure 1-7. The first example shows a spatial near-miss case, two road objects are nearly colliding with each other. The second example shows a temporal near-miss case, the front white car suddenly stopped in the middle of the intersection, forcing a sudden brake for the back white car.

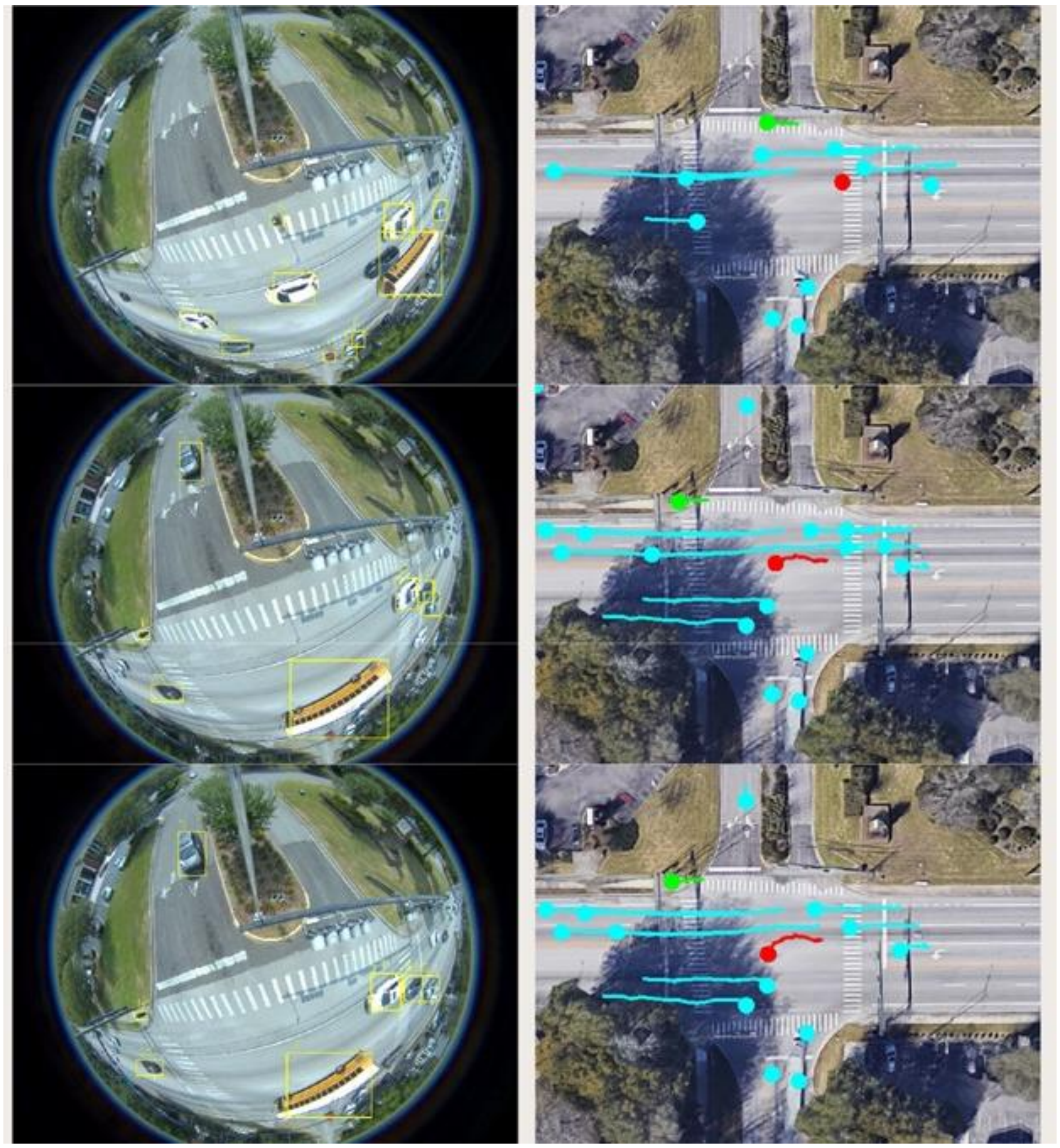


Figure 1-6. Road object trajectories projected on satellite map. Left: frame sample of tracking. Right: trajectories after mapping

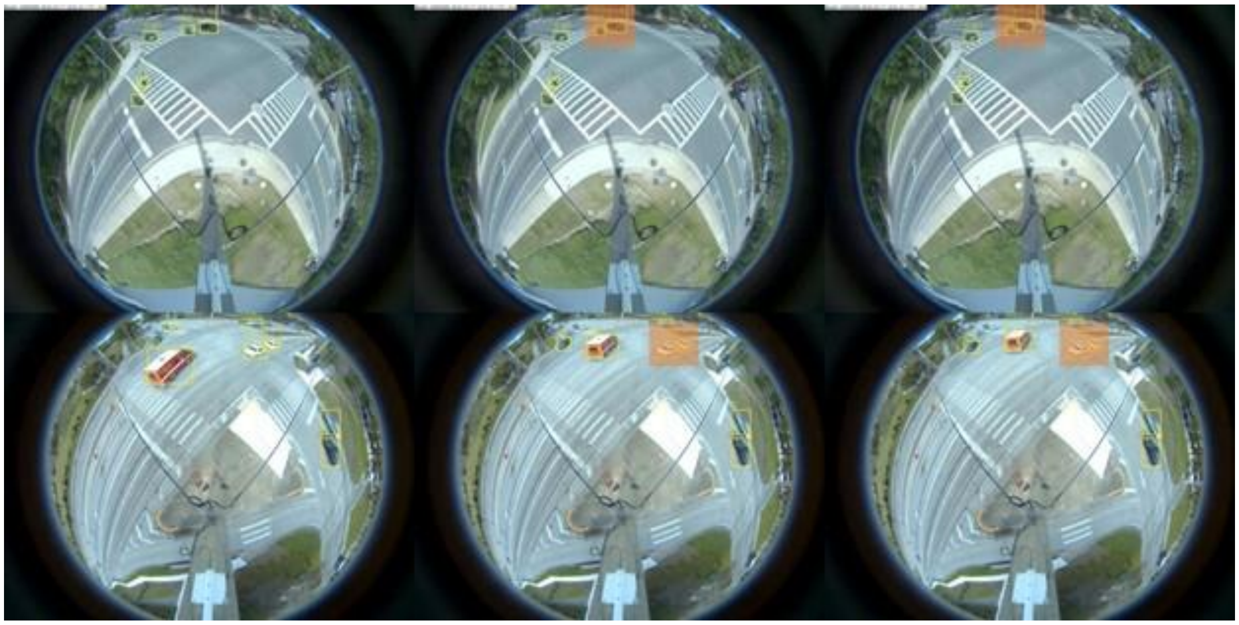


Figure 1-7. Examples of two types of near-miss detected. Top 3 images: spatial near-miss, a motorbike and a car are colliding. Bottom 3 images: temporal near-miss caused a sudden brake.

1.4. Quantitative Evaluation

We present a quantitative evaluation of the overall performance of our proposed method in terms of speed performance, improvement object speed measures based on mapping, and the precision and recall for each subtask of the pipeline.

1.4.1. Computational Requirements

We present speed performance for the tested methods in Table 1-1. The fisheye video resolution is 1280 × 960, and our implementation for thin-plate splines takes 10 seconds for one-to-one correspondence mapping for 1,228,800 points. After getting mapping point-sets, all video processing experiments have been performed on a single GPU (NVIDIA TITAN V). The GPU-based Simple Linear Iterative Clustering (SLIC) segmentation (Achanta et al., 2012) has excellent speed performance, running at 400 fps with fisheye videos. The overall pipeline of our methods (object detection, multiple object tracking, and near-miss detection) have achieved real-time performance of about 40 fps, which has advantages for real traffic surveillance and near-miss detection for large-scale daily video data.

Table 1-1. Quantitative evaluation of speed performance on central processing units (CPUs) and graphics processing units (GPUs)

Methods	CPU/GPU	Speed
TPS mapping	CPU	10 s
SLIC segmentation	NVIDIA TITAN V	400 fps
Overall pipeline	NVIDIA TITAN V	40 fps

Table 1-2. Quantitative performance of object detection and multiple object tracking.

Methods	TP	FN	FP	Precision	Recall	F1-score
Object Detection	7649	102	82	0.98940	0.98684	0.98812
Multiple Object Tracking	7540	483	314	0.96002	0.93980	0.94980

Table 1-3. Quantitative comparison of near-miss detection between non-mapping and calibration+TPS-based method.

Methods	Intersection	TN	FP	Specificity (TNR)	Fall-out (FPR)
Non-mapping based (Baseline)	intersection 01	2869	32	0.98897	0.01103
	intersection 02	2659	162	0.94257	0.05743
Calibration + TPS mapping	intersection 01	2895	6	0.99793	0.00207
	intersection 02	2818	3	0.99894	0.00106

* TN: True Negative, FP: False Positive, TNR: True Negative Rate, FPR: False Positive Rate.

1.4.2. Trajectory and Near-miss Detection

To evaluate our prediction of object representation and near-miss, we compared predicted detection with ground truth at frame level. A true positive is that a prediction–ground-truth detection pair has an Intersection-over-Union (IOU) score which exceeds a predefined threshold (e.g., 0.7) for a track that is correctly associated. A true negative means no prediction and no associated ground truth. A false positive is that a prediction had no associated ground truth. A false negative is that a ground truth had no associated prediction. The true negative rate (TNR) is referred to as specificity, and the false positive rate (FPR) is referred to as fallout. The specificity, fallout, precision, recall, and F1-score are defined as:

$$TNR = \frac{TN}{TN + FP} = 1 - FPR \tag{1.1}$$

$$Precision = \frac{TP}{TP + FP}; Recall = \frac{TP}{TP + FN} \tag{1.2}$$

We compute object speed information based on trajectories by converting pixels to actual meters and frame intervals to seconds. Figure 1-8 shows an example of the comparison of computed object speed information where a car is approaching the intersection with speed decreasing from 60 km/h to 20 km/h and then increasing to 30 km/h. With non-mapping methods, object speed computing suffers from fisheye and perspective distortion and yields inaccurate results. We also present accuracy evaluation for object detection and multiple objects (cosine metric learning) in Table 1-2. As real near-misses are rare even in a week's worth of video from two cameras, it is more reasonable to exam specificity (selectivity or true negative rate) and fallout (false positive rate) for near-miss detection. In Table 1-3, we present the comparison of non-mapping based detection and calibration+TPS mapping based detection in term of true negative rate (TNR) and false positive rate (FPR). The quantitative evaluation demonstrated the overall effectiveness of our proposed method for near-miss detection in large-scale fisheye traffic videos.

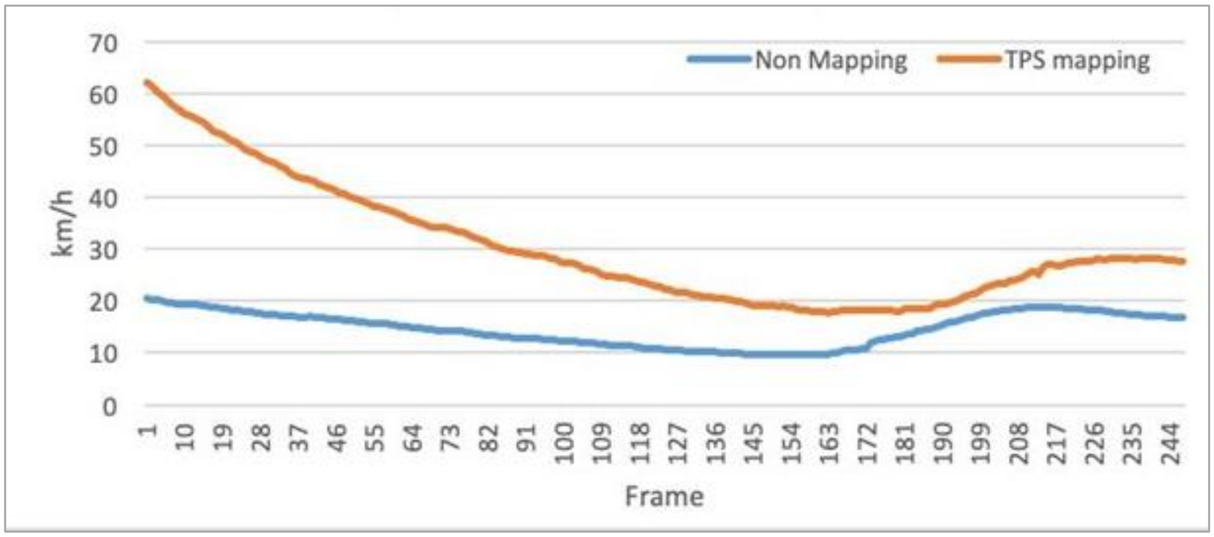


Figure 1-8. Comparison of computed object speed information between non-mapping and proposed methods

1.5. Unwarping a Fisheye Image

In this section, we describe the various transformations we apply to flatten a fisheye image from a single camera and also how we stitch two images from different cameras placed at the same intersection to come up with a combined image.

1.5.1. Correcting Distortions from a Single Camera

There are two types of distortions predominant in images from fisheye cameras, radial and perspective distortions, which come into play as we try to map a fisheye image to the corresponding top-down image (as captured by a satellite, as in a Google Maps image). In this section, we discuss the transformations used to correct these distortions.

The first step in distortion correction is to find the distortion parameters. There are two types of distortion parameters: intrinsic and extrinsic parameters. The intrinsic parameters relate to the camera properties, including its focal length and optical center, while the extrinsic parameters relate to the camera position and rotation with respect to the vehicle reference frame. Intrinsic parameters are used to create the camera matrix, which is given by Equation 1.5.

Let (x_c, y_c, z_c) be the camera reference frame coordinates of an object at location P. To compute the location (u, v) of the object in a 2D unwarped image, we apply a series of two transformations. First, we compute (x_0, y_0) using Equations 1.3 and 1.4, to address the extrinsic camera parameters. Next, we apply Equations 1.5 and 1.6 on (x_0, y_0) to address the intrinsic camera parameters. Finally, we obtain (u, v) as coordinates of P in the unwarped image.

$$\theta_{distorted} = \theta \left(1 + k_1 \theta^2 + k_2 \theta^4 + k_3 \theta^6 + k_4 \theta^8 \right) \tag{1.3}$$

$$x' = \left(\theta_{distorted} / r \right) a; \; y' = \left(\theta_{distorted} / r \right) b \tag{1.4}$$
$$where\ a = x_c / z_c,\ b = y_c / z_c,\ r^2 = a^2 + b^2,\ and\ \mathrm{atan}(r)$$

$$Camera\ Matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

$$u = f_x (x' + \alpha y') \, c_x \; ; \; v = f_x (y') \, c_y \tag{1.6}$$

In the above equations, k_1, k_2, k_3, k_4, $\theta_{\text{distorted}}$, and θ are the extrinsic parameters. $\theta_{\text{distorted}}$ and θ are the distorted and original angles of point P with respect to the camera-reference frame; x_0 and y_0 are distorted pinhole projected point-coordinates; f_x and f_y are focal lengths; c_x and c_y are the coordinates of the optical centers.

We take pictures of a checkerboard with the fisheye camera as shown in Figure 1-9 and run training algorithms to estimate the parameters k_1, k_2, k_3, and k_4 from the equations above. The idea is to keep the straight lines in the checkerboard as straight lines in the final image. By substituting the values of coordinates of the corners detected from images and the corresponding corners needed in the final image, we get the approximate value for the parameters. We used the calibrate function available in the OpenCV library to accomplish this. Once the parameters are obtained, we use them directly in the Equations 1.3 through 1.6 to transform the traffic video captured by a fisheye camera. Figure 1-10 shows the original image and the image after the transformations are applied.

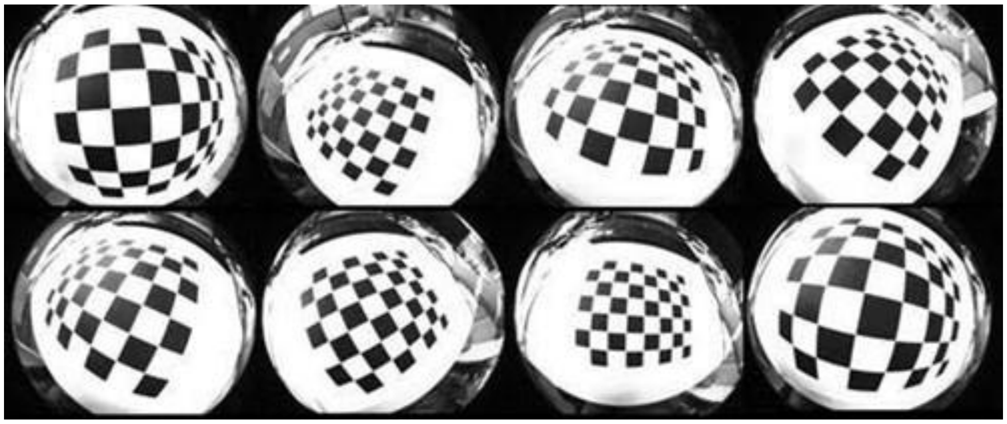


Figure 1-9. Samples of distorted images of a checkerboard taken with a fisheye camera

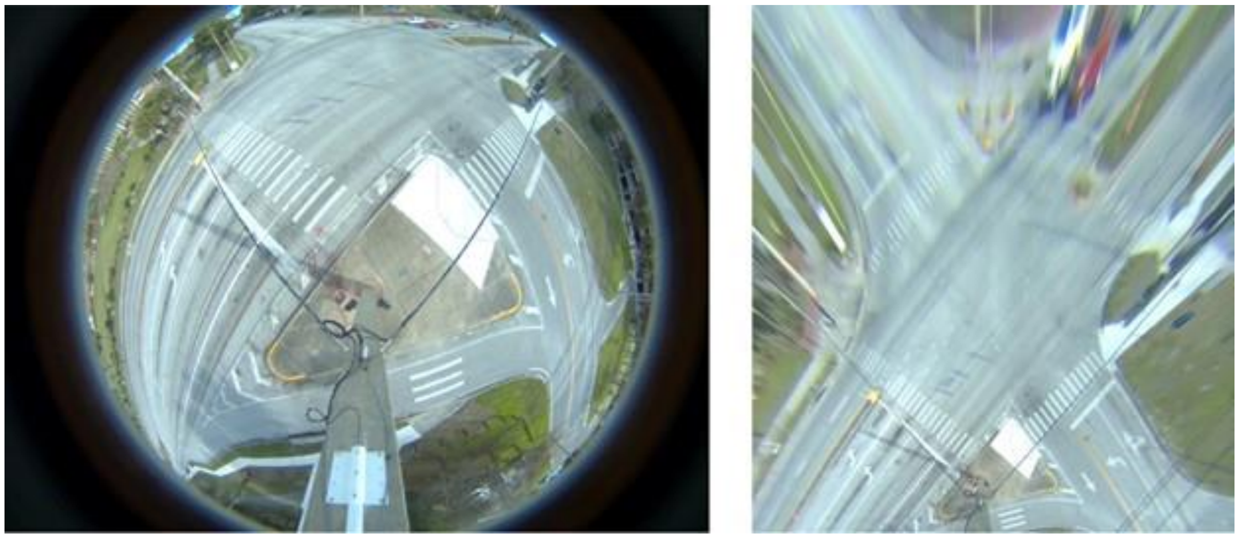


Figure 1-10. Left: Original fisheye image; Right: the corresponding distortion corrected image

At any given traffic intersection, a fisheye camera is usually installed at one of the four corners of the intersection. Though the ideal location for camera placement would be above the center of the intersection, it is not possible to place a camera physically there because of accessibility issues. Having a camera at one corner results in the optical center being located directly below the camera at that corner. This unfortunately causes massive distortion of regions in the intersection that are relatively far away from the camera. Because our focus is on the whole intersection, we experimented with different values of the optical center (c_x and c_y) that is closer to the center, and this improved the quality of the image. The image on the right-hand side of Figure 1-10 was generated after applying corrected optical center coordinates.

After applying correction for distortions due to properties of a fisheye camera, we correct additional distortions using the Thin Plate Spline (TPS) algorithm (Sprengel et al., 1996). Using the Google Maps image in the middle of Figure 1-11 as a reference, we identify about 12 point-wise mapping coordinates between the intersection images in the left and middle of Figure 1-11, and using these coordinates obtain the TPS approximation function. The results obtained after TPS transformation look very similar to the Google map, as is shown in the right of Figure 1-11.

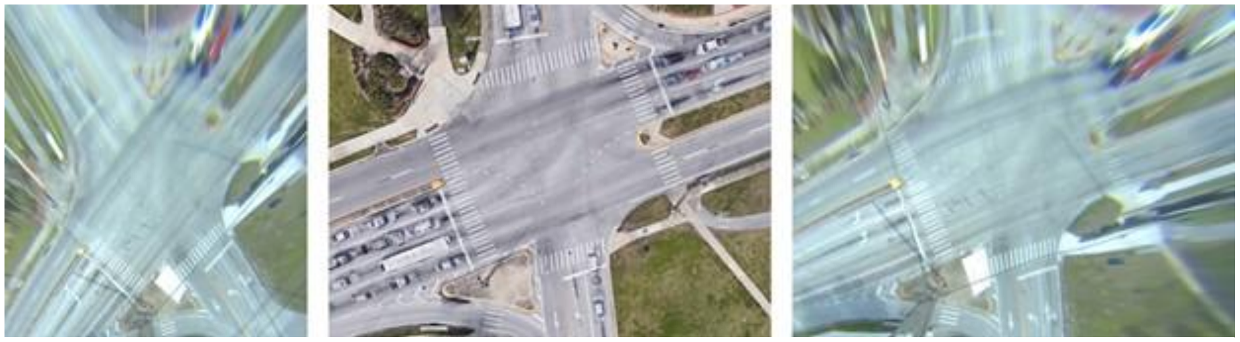


Figure 1-11. Left: Image after distortion correction. Middle: Google-Map reference. Right: Image after TPS transform

An additional step is required to correct distortions due to the height of a vehicle. The procedure adapted to correct this problem is called lane fitting. The main principle we use in lane-fitting is that if the ground is mapped properly, the distortion created by any vehicle is in a direction away from the camera and its distortion length is a polynomial function of the distance of the vehicle from the camera. Different vehicles have different polynomial behavior with distance mainly due to their different heights. To approximate the average behavior, we use the path data obtained from the vehicle detection algorithm of video processing and the corrected paths obtained using TPS. We create plots for all the tracks where the x-axis represents the distance of tracked point from camera and the y-axis represents the distance from the true point. A true point exists for every tracked point and it is defined as the point of intersection between a line joining the tracked point to camera and the drawn true path. Here, true path refers to a straight line that is manually drawn through the lane markings on the road. True paths are calculated only for the vehicles going straight through intersection. After getting a sufficient number of such points, a polynomial of degree 3 is fitted and the transformation function is used to transform all the points. This algorithm of lane-fitting solves the problem of track distortion due to large vehicles and also reduces the speed of vehicles which in many cases always increases when a vehicle moves away from the camera. As we will discuss below, the lane-fitting algorithm is useful when we stitch vehicle tracks from multiple cameras, since lane-fitting decreases the distance between the tracks from different cameras for the same vehicle.

1.5.2. Stitching Images from Multiple Fisheye Cameras

Large intersections generally have two or more cameras installed at different corners of intersection. Each camera can cover only part of intersection with good quality. Stitching of these camera videos is necessary to get a panoramic view of the intersection in the final mapped video. For blending the two images, we use the basic weighted-alpha-based blending (Fitzgibbon, 2001). The process is shown below in Figure 1-12. For merging the trajectories, we use time-synchronized tracking data from all the cameras. Considering the situation where vehicle is tracked by all the cameras placed at different locations of the intersection, the first step is to generate unwarped trajectories from each camera into the same Google Maps image using the unwarping algorithms as described above. However, the trajectories of the same vehicle as recorded by multiple cameras may not exactly align as we have height distortion even though it is somewhat mitigated by the lane fitting algorithms. We solve this problem by using registration and confidence score weighted mapping.

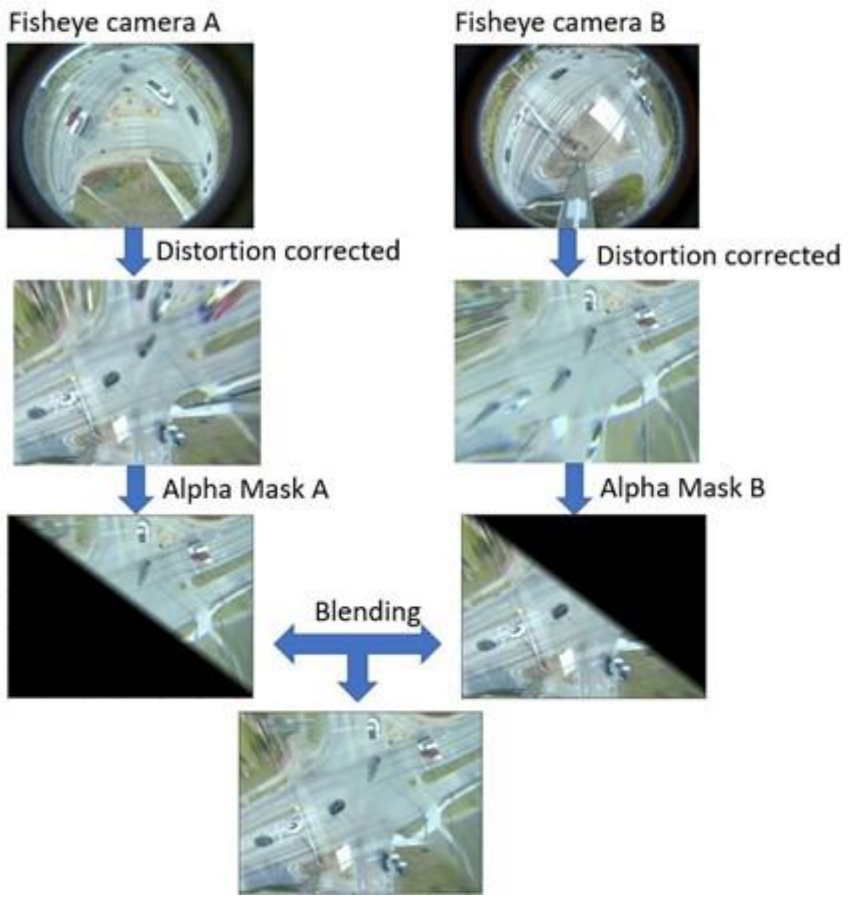


Figure 1-12. Pipeline of multicamera fisheye image

During the registration stage, the trajectory of a vehicle from one camera is matched to trajectories of the same vehicle from the other camera(s). A simple method to pick a trajectory in the image produced by one camera and to compare this trajectory with the time synchronized coordinates of every vehicle in the other cameras. The comparison is done by finding the distance between the coordinates of the picked trajectory to the corresponding coordinates of the other trajectories that occur during the same time and are captured using the other cameras. For a two-camera intersection, the distance of every track of the second camera in the given time window, is measured against a selected track of the first camera. The distance values are stored, and the trajectory of the second camera with least distance with the selected trajectory from the first camera are matched.

Using the trajectories of the same vehicle from different cameras, the average weighted path is calculated by using weights based on the distance of a track coordinate from a camera. Theoretically, the more the distance of camera the less will be tracking accuracy. Based on this principle, the weighted point is calculated using multiple points based on the ratio of distance of the points to the cameras.

1.6. Conclusions

In this work, we have focused on video processing using fisheye cameras at intersections. Since the cameras are still mounted at intersections, video processing was oriented toward traffic entering and exiting the intersection and on the pedestrians (and cycles etc.) present in the field of view. The video was rectified from the fisheye wide angle view and registered to Google map coordinates using a thin-plate spline warping. This allowed us to visualize traffic and pedestrians in pseudo-rectilinear coordinates (stemming from the warping and registration steps). Future work will focus on improved turn movement counts, pedestrian tracking etc. in a variety of weather conditions (since these have only been partially explored) so far. The fisheye geometry affords multiple perspective rectilinear mappings and exploration of these views will also be part of future work.

Chapter 2 – Develop Algorithms and Software for Nighttime Camera Video Processing-based Traffic Analysis

2.1. Introduction

The advent of intersection cameras as sensors have added a dimension to intersection event monitoring and analysis. Fisheye cameras are more likely to be the choice for traffic monitoring at intersections due to its wide angle, which makes the setup of monitoring system in intersections easier. Because of that, demands for video processing on fisheye videos has increased. While induction loop detectors can detect the presence of vehicles at certain parts of the intersection (where these are installed), they are unable to capture the dynamics of vehicle to vehicle of vehicle to pedestrian interactions. A drawback of the cameras is that they need the intersection to be well lit for the vehicles and pedestrians to be visible. This is the main topic for this task. We have analyzed an intersection West University Avenue and 13th Street, here in Gainesville, FL, for different lighting conditions using our advanced multi-object tracking algorithm and present the results here. We have also studied the performance of our algorithm on two other intersections on West University Avenue and one intersection of NW 23rd Avenue in Gainesville and two intersections in Lake Mary, Orlando, to evaluate the conditions under which we get reasonable performance. The intersections in Lake Mary Orlando are Lake Mary Blvd. and Rhinehart Road and Postal Distribution Center and Rhinehart Road. All studied intersections are shown in Figure 2-1.

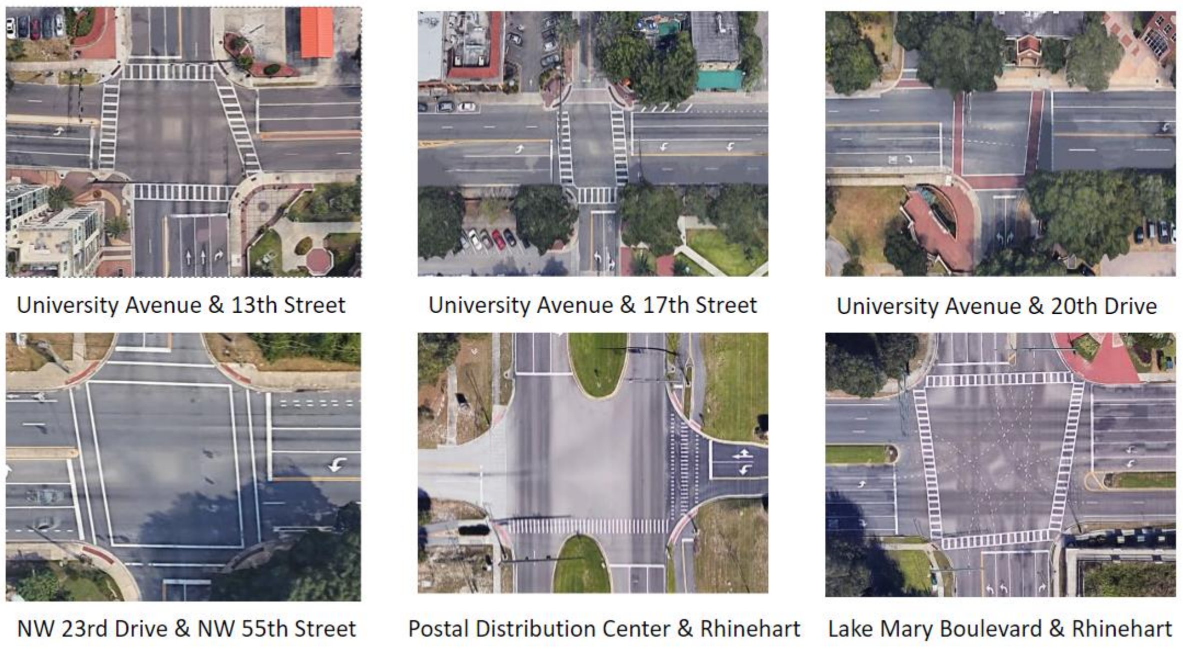

University Avenue & 13th Street
University Avenue & 17th Street
University Avenue & 20th Drive
NW 23rd Drive & NW 55th Street
Postal Distribution Center & Rhinehart
Lake Mary Boulevard & Rhinehart

Figure 2-1. Aerial view of each subject intersection

2.2. Algorithmic Modifications

Nighttime video processing, near-miss detection, traffic analysis, traffic rule violations, and other similar applications need a stable and fast method for multi-object tracking in the intersection. Multi-Object tracking is a common computer vision problem that may be divided into object detection and object tracking. Various models can be used for object detection, such as You Only Look Once (YOLO), Region-based Convolutional Neural Network (R-CNN), and CenterNet. The differences between the three algorithms, YOLO, R-CNN, and CenterNet, are detailed (Liu et al., 2020). Briefly, R-CNN is a two-stage object detection method, and YOLO is a one-stage method. Both YOLO and R-CNN are anchor-based methods, and CenterNet is an anchor-free method. All those models have a reasonably good performance.

Performance may be measured as inference speed or as Average Precision (AP). AP is derived from precision, and recall is represented as the area under the precision-recall curve. In general, the faster the inference speeds, the lower the AP. For example, YOLO is the fastest inference speed because it is a one-stage method. It is unclear how YOLO can translate to good performance on datasets with many objects per image. R-CNN has slow inference speeds comparatively but better localization and hence AP. CenterNet also has slower inference speeds, but good AP. DeepSORT is the most popular and widely used algorithm due to its speed and easy combination with object tracking. However, the radial distortion in the fisheye camera makes the task, especially object tracking, much harder than in the regular rectilinear videos.

The radial distortion mainly affects object tracking in three different ways. First, the shapes and directions of objects will constantly be changing, making the feature vector less functional in tracking objects. This issue is mainly due to the information lost in the fisheye camera compared to the standard camera, so we cannot easily solve this issue. Second, the motion prediction of the Kalman Filter cannot process the complicated motion in a fisheye camera. An object in a fisheye video can become larger then smaller, rotate clockwise then counterclockwise, while moving faster and then slower. These motion characteristics are an artifact of the camera itself and not the object. This wide variation exceeds the range of parameter values in the Kalman Filter. In this task, we will present a method to enhance the motion prediction by assisting the Kalman Filter with fisheye space transformed to aerial or bird-eye space.

Our work is based on implementing enhanced DeepSORT for multi-object tracking, which uses optical flow to improve motion prediction. The algorithm utilizes the Kalman Filter in two stages. During the assignment on motion and feature, we apply the space transform on the bounding boxes before calculating motion distance. By introducing the bird-eye space, the movement is much more simplified than in the original fisheye space, due to the stabilization of size and speed. We use the intersection over union (IOU) tracker which only considers the overlap of localization bounding boxes between the current and previous frames in the video. For this tracker, we let the detected box from fisheye as input to the algorithm.

The quality of results with this approach will thus depend on the space transformation which must be accurate enough while the unwarped fisheye image is still highly distorted. Since the fisheye camera is fixed, we use the satellite map as the baseline to achieve this transformation. The mapping becomes nearly perfect after applying a TPS on a dozen manually selected matching points. While we map the background accurately, we also map the bounding boxes in the aerial space and modified the Kalman filter to use the aerial bounding boxes instead of the bounding boxes in the fisheye space. We reuse the bounding box prediction functionality of the original Kalman filter which helps us in the following ways.

1. Filling in data for missing frames
2. Doing IoU matching in DeepSORT algorithm to track missing objects.

Our modified algorithm works well on the fisheye camera multi-object tracking in real-time, both in the daytime and nighttime scenarios. Our experiments indicate that it increases the tracking accuracy in the fisheye video with nearly no cost on time. It works well on the datasets from the City of Gainesville and Orlando, including multiple fisheye cameras in different intersections. After the modification on motion detection, the high order tracking accuracy (HOTA) score of tracks has increased a factor of 10 on average among ten other videos from various intersections and times.

We tried other approaches for processing the nighttime videos based on literature, which are listed below:

1. Tracking headlight (front lights and back lights):
   We experimented with this approach in a small prototype to monitor the headlights of vehicles. In this method the headlights are captured and analyzed. This approach was not fit for our software because we use fisheye cameras and introducing sensitivity to the headlights dimensions (length/width) or distance between the two lights to determine if these are from the same vehicle, produced erroneous results due to radial distortions present in the fisheye video itself.
2. Low-light image enhancement:
   In this technique we enhance the contrast of the objects to better identify them. This technique had comparable performance as treating the non-enhanced video through the YOLO detector and DeepSORT tracker.

Compared to our previous detection rate of about 30-50% of vehicles at night in a well-lit intersection, our current algorithm has over 90% detection accuracy for well-lit intersections, and about 80% accuracy for moderately or low-lit intersections for vehicles. For pedestrians however, the detection is poor (such as even 0% sometimes) because of the small size and very little contrast with background at night. Pedestrian detection is a challenge for the earlier as well as the current and enhanced version of our video processing software. We do have to use other technology for pedestrian detection at night. LIDAR is a promising alternative to fisheye camera for pedestrian detection at night. The processing speeds have improved with these modifications as well. For example, the current processing rate is 35-90 frames per second (FPS), whereas the previous processing was in the range of 10-20 FPS.

2.3. Experiments

We describe certain specific examples of processing early morning and late-night videos at West University and 13th Street. We will also present examples from other intersections.

2.3.1. West University and 13th Street in Gainesville, FL

We counted the number of cars passing the West University and 13th Street intersection and compared it with the number of cars tracked by the software to check the detection and tracking functionality at night time (Figures 2-2 to 2-4). For the studied interval, the 22 out of 25 vehicles were detected fine at around 11 PM. The cars that pass over the far side of intersection with respect to the camera were missed. Most of the pedestrians were also missed because of the poor visibility and contrast. It would be required to use specialized nighttime cameras to address these issues.

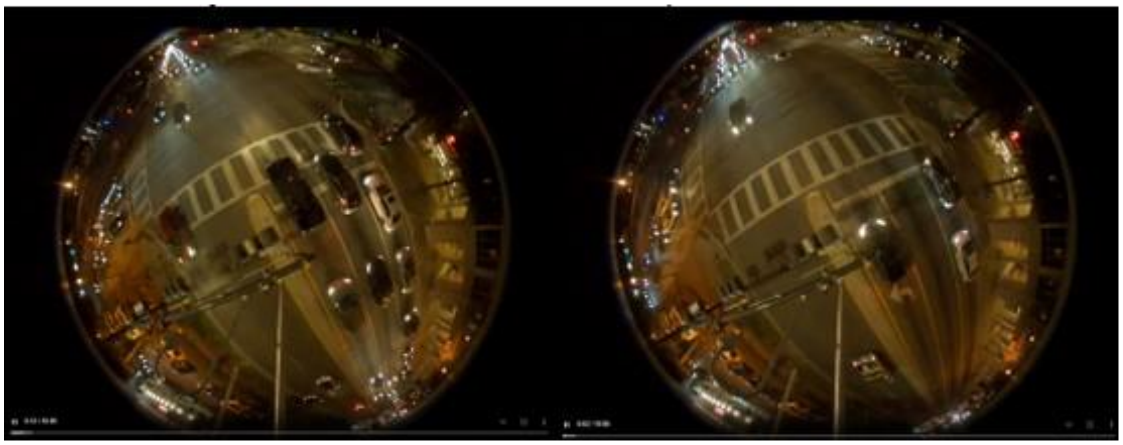


Figure 2-2. The West University Avenue and 13th Street intersection at 8 PM (left) and 9 PM (right)

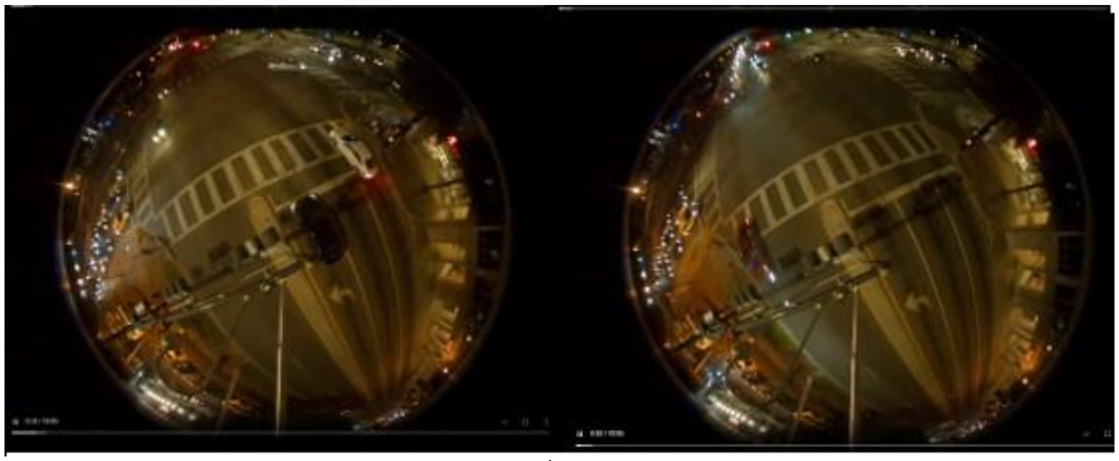


Figure 2-3. The West University Avenue and 13th Street intersection at 10 PM (left) and 11 PM (right)

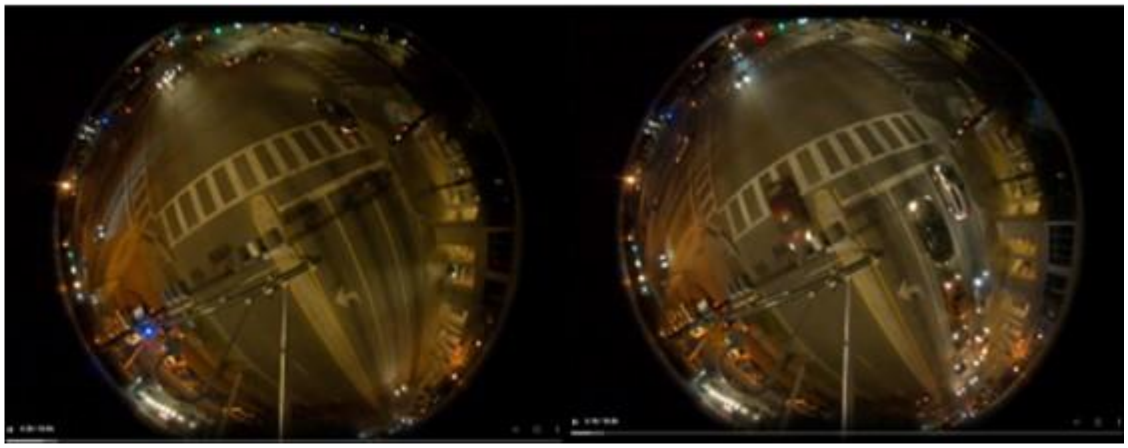


Figure 2-4. The West University Avenue and 13th Street intersection at 6 AM (left) and 7 AM (right)

2.3.2. West University and 17th Street, Gainesville, FL

For the West University and 17th intersection, we studied the time interval at around 11:50 PM (Figures 2-5 and 2-6). There were nine vehicles, and all of them were detected and tracked. This intersection however has many pedestrians at that time, and these could not be tracked from the low contrast fisheye images.

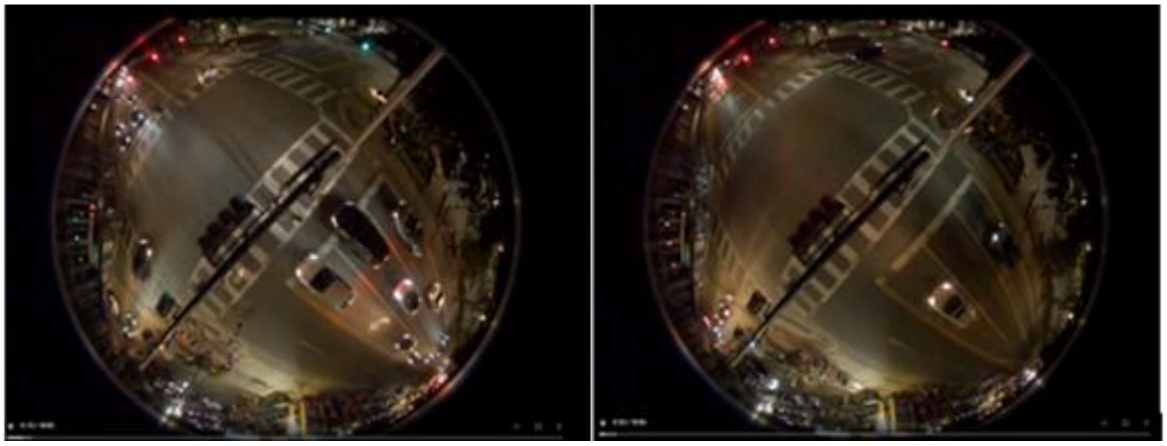


Figure 2-5. The West University Avenue and 17th Street intersection at 8 PM (left) and 9 PM (right)

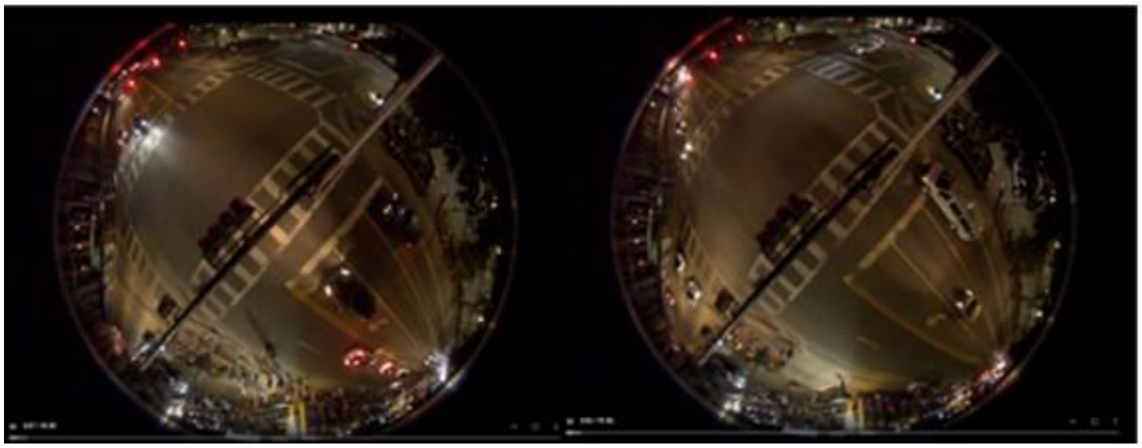


Figure 2-6. The West University Avenue and 17th Street intersection at 10 PM (left) and 11:50 PM (right)

2.3.3. West University and 20th Drive (Gale Lemerand), Gainesville, FL

At this intersection for the select time interval at around 8 PM, we found 37 vehicles (Figures 2-7 to 2-8). Thirty-three could be detected and tracked.

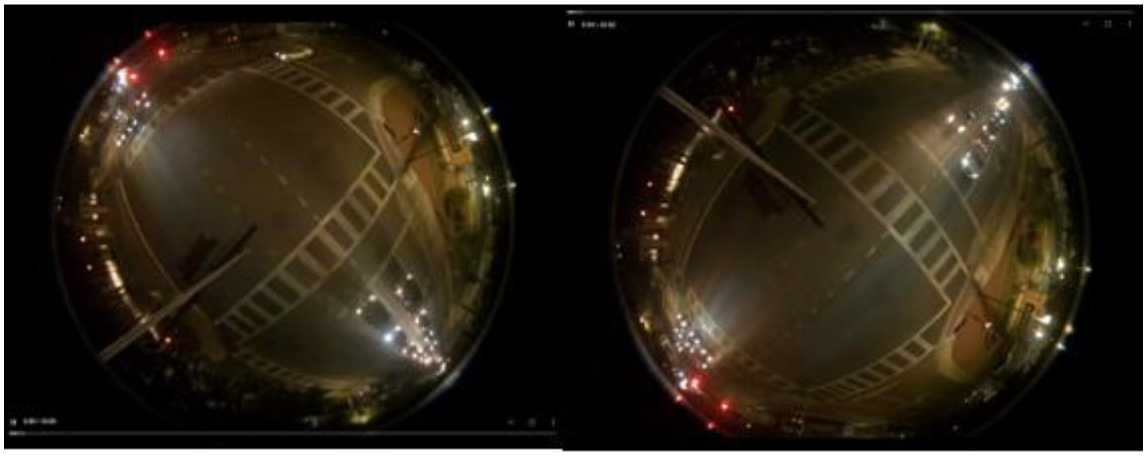


Figure 2-7. The West University Avenue and 20th Drive (Gale Lemerand) intersection at 8 PM (left) and 9 PM (right)

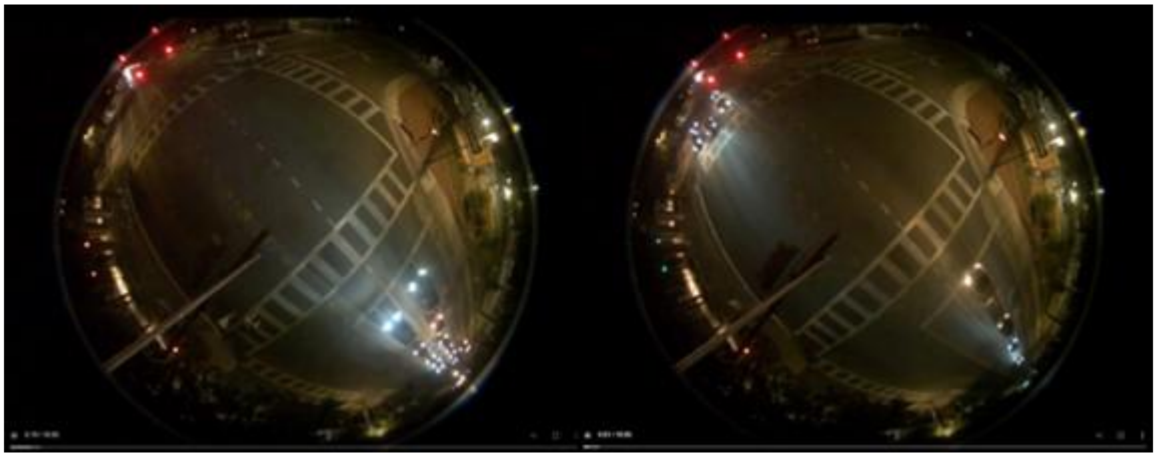


Figure 2-8. The West University Avenue and 20th Drive (Gale Lemerand) intersection at 10 PM (left) and 11 PM (right)

2.3.4. NW 23rd Avenue and NW 55th Street, Gainesville, FL

This intersection is in close proximity to a high school and experiences unique traffic patterns during drop-off and pickup times on weekdays (Figure 2-9). This is an isolated intersection and is not a part of a corridor. We counted the traffic at this intersection and found that at around 7 AM there were 23 vehicles moving through the intersection; our video processing algorithm could track 18. The detection and tracking performance on this intersection is expected to improve following the addition of more annotations.

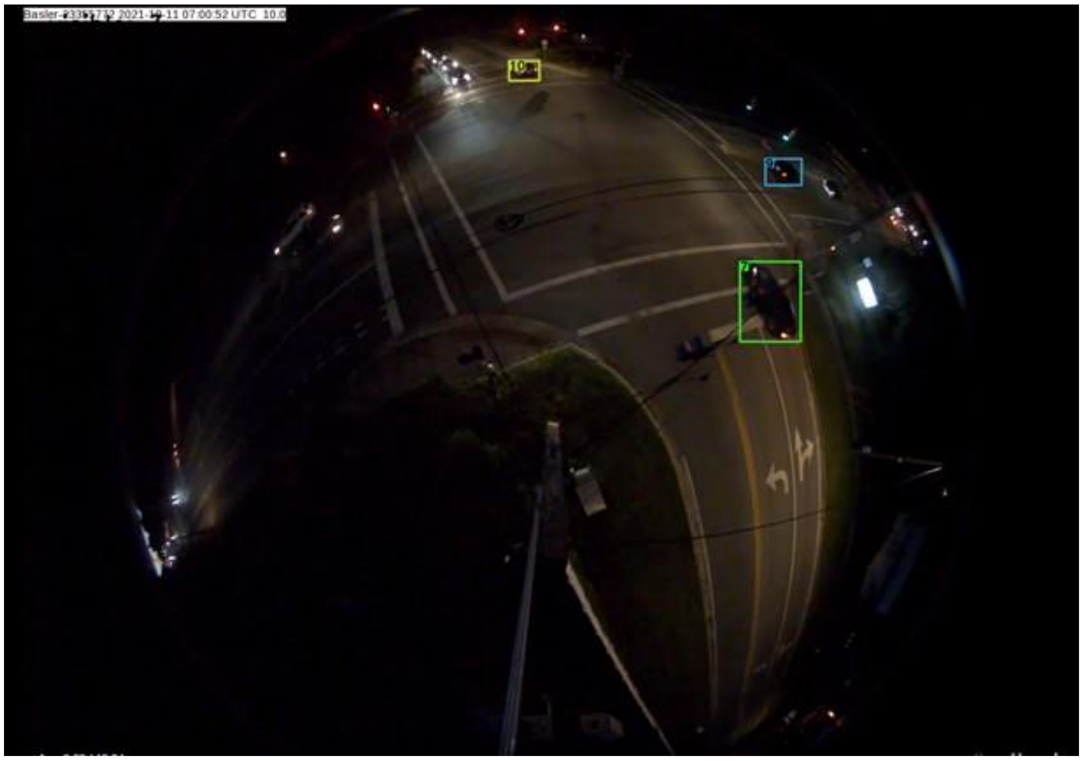


Figure 2-9. Intersection at the crossing of NW 23rd Avenue and 55th Street, in Gainesville at 7 AM

2.3.5. Postal Distribution Center and Rhinehart Road, Orlando, FL

This intersection is in Seminole County (Figure 2-10). The intersection was considerably dark, and we evaluated the performance of our new video processing algorithms. At around 6 AM, we could detect five out of six vehicles at the intersection. The vehicle that could not be detected passed along the darker side of the intersection.

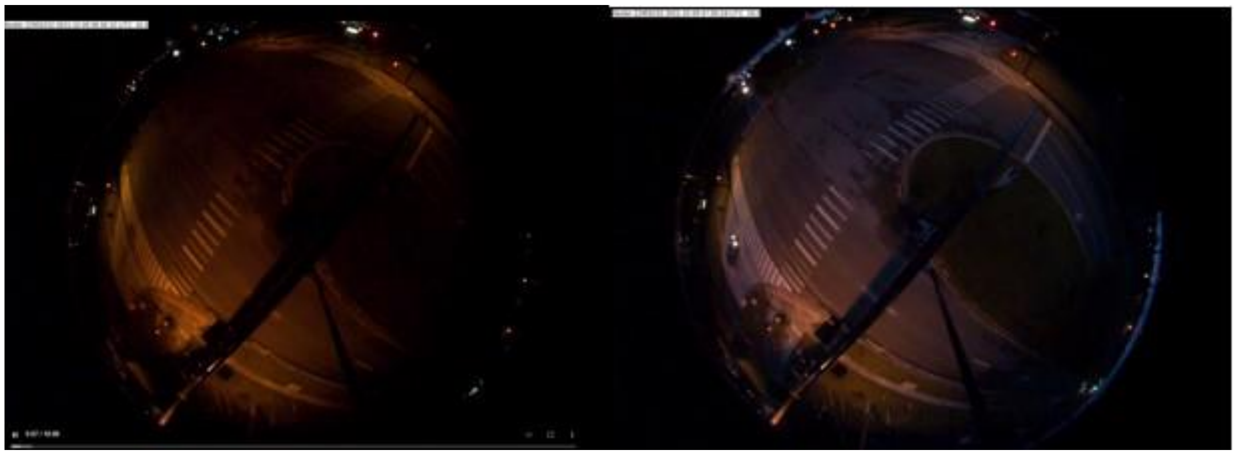


Figure 2-10. Postal Distribution Center and Rhinehart intersection in Seminole County, at 6 AM (left) and 7 AM (right)

2.3.6. Lake Mary Boulevard and Rhinehart Road, Orlando, FL

This is the second intersection in Seminole County that we processed (Figure 2-11). At around 7 AM, our new video processing algorithms could detect 17 out of 20 vehicles that passed across the intersection for the sample time interval.

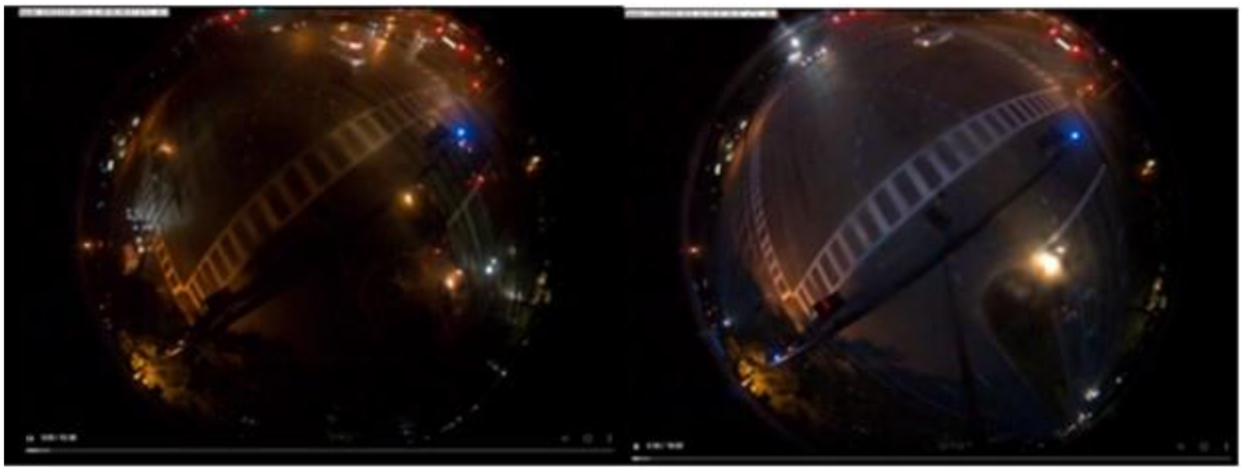


Figure 2-11. Lake Mary Boulevard and Rhinehart Road intersection at 6 AM (left) and 7 AM (right)

Table 2-1 summarizes the observed counts of vehicles in video clips and the corresponding vehicles tracked by our new video processing algorithm for the different intersections. Vehicle detection works best when the vehicles are near the camera. Vehicles far from the camera and away from street light are not detected. Pedestrian counts have been excluded because fisheye cameras cannot detect pedestrians at night satisfactorily.

Table 2-1. Vehicle counts at different intersections during the night and early morning.

Intersection	Time	Actual count	Detected count	Percentage
West Univ & 13th Street, Gainesville	09:00 PM	65	55	84
West Univ & 13th Street, Gainesville	11:00 PM	25	22	88
West Univ & 17th Street, Gainesville	10:00 PM	44	44	100
West Univ & 17th Street, Gainesville	11:50 PM	9	9	100
West Univ & 20th Street, Gainesville	08:00 PM	37	33	89
West Univ & 20th Street, Gainesville	10:00 PM	35	35	100
NW 23rd Ave & NW 55th St, Gainesville	07:00 AM	23	18	78
PO & Rhinehart Rd, Orlando	06:00 AM	6	5[1]	83
PO & Rhinehart Rd, Orlando	07:00 AM	15	3[2]	20
Lake Mary Blvd & Rhinehart Rd.	06:00 AM	10	3[3]	30
Lake Mary Blvd & Rhinehart Rd.	07:00 AM	20	17[4]	85

1. These vehicles were taking East Bound Left and were relatively better lighted
2. These vehicles were North- and South-bound with very poor lighting
3. Very poor lighting at the intersection
4. Better lighting by 7 AM EDT.

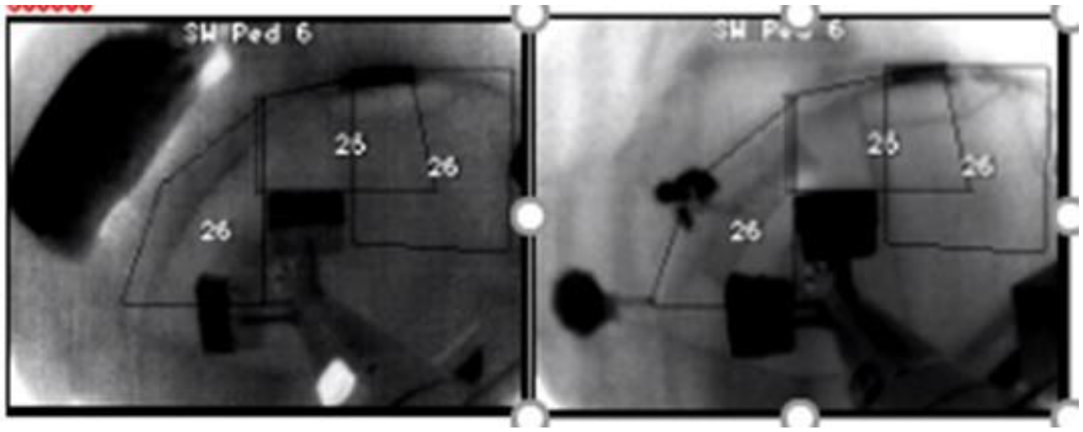


Figure 2-12. FLIR camera images from W University Ave and 17th Street. The left image captures a vehicle, while the right one captures a pedestrian

FLIR Camera Images – Figure 2-12 shows images from a FLIR camera. The numbers in the image correspond to the detector channels. The vehicle detectors also act as passive pedestrian detectors. At the West University Avenue and 17th Street intersection, which is shown in the Figure 2-12, the detectors are numbered between 21-28. We have not evaluated the FLIR cameras to be able to draw a comparison between these and the fisheye cameras.

2.4. Conclusion

We conclude that our new algorithms perform reasonably well on videos collected using fisheye cameras during nighttime if the intersection is moderately lit. Among the updates, we enhanced our detection algorithm to YOLOv4 and our tracking algorithm to fast multi-object tracking that improves detection and tracking performance for nighttime as well as daytime videos. We presented the results of tracking based on arbitrarily selected small intervals of time.

Compared to our previous detection rate of about 30-50% of vehicles at night in a well-lit intersection, our current algorithm has over 90% detection accuracy for well-lit intersections, and about 80% accuracy for moderately lit intersections for vehicles. For low-lit intersections, if the vehicles cannot be seen, it is not possible to detect the vehicles to a reasonable degree of accuracy. For pedestrians however, the detection is poor (such as even 10% or lower) because of the small size and very little contrast with background at night. Pedestrian detection is a challenge for the earlier as well as the current and enhanced version of our video processing software. We do have to use other technology for pedestrian detection at night.

Intersections that are very dark with little to no visibility will require more lights or must be fitted with LIDAR cameras or with thermal sensitive FLIR cameras targeted at specific regions that need to be monitored. We studied the use of LIDAR for detection of pedestrians and vehicles at night and have seen promising results which may be found in the Task 3 report for LIDAR point clouds. LIDAR technology can be used during the day and at night because of an active illumination sensor, which is not affected by light variations such as brightness and the darkness of daylight.

Thus, it is recommended to have both fisheye cameras and LIDARs at critical intersections. Since the cost of sensor devices and storage is going down quickly and this trend also applies to computing resources; hence, it should be a viable solution to have fisheye cameras and LIDARs at critical intersections.

Chapter 3 – Develop and Test LIDAR Processing for Pedestrian Counts and Movement Patterns

3.1. Background Statement

The rapid changes in growth of exploitable and, in many cases, open data have the potential to mitigate traffic congestion and improve safety. Despite significant advances in vehicle technology, traffic engineering practices, and analytics based on crash data, the number of traffic crashes and fatalities are still too many. Many drivers are frustrated due to long (but potentially preventable) delays at intersections. The use of video/LIDAR processing, big data analytics, artificial intelligence, and machine learning can profoundly improve the ability to address these challenges. The collection and exploitation of large data sets is not new to the transportation sector. However, the confluence of ubiquitous digital devices and sensors, significantly lower hardware costs for computing and storage, enhanced sensing and communication technologies, and open-source analytics solutions have enabled novel applications through specific uses and by combining with other data sources. The latter may involve insights into otherwise unobserved patterns that may positively influence individuals and society.

Intelligent transportation systems require the use of interactions between road users and infrastructure. Dedicated short-range communications (DSRC) using radio, Wi-Fi, or cellular technologies can enable such interactions at signalized interactions. By using DSRC effectively, the infrastructure systems can provide information to the users about the interactions as well as using the road users as probes to create a vignette of local and network level traffic patterns and usages. The forthcoming Gainesville Signal Phasing and Timing (SPaT) Trapezium project will deploy and test connected vehicle technologies and applications along four roads forming a trapezium surrounding the University of Florida main campus, as shown in Figure 3-1. The goal of the project is to improve travel time reliability, safety, throughput, and traveler information. Approximately 45 roadside units will be installed in this project. The roadways and intersections along and within this "Trapezium" and bus routes serving this area will constitute the fundamental real-world test bed for this study. Most of the signalized intersections of interest to this study have live CCTV RTSP feeds streamed at 30 frames per second, HD quality (720p or 1080p), pan, tilt, and zoom capabilities and video detection for stop bar/presence and traffic counting. At select intersections, multimodal video detection for pedestrians, bicycles, and vehicles (using Iteris Vantage Live, Smart Cycle, and PedTrax) and fisheye video detection with motion tracking and vehicle classification capabilities are planned for installation. ATC controllers can provide signal-timing history at decisecond resolutions. Thus, a variety of video feeds and signal timings are available that will be used for ratification of the methods developed in this proposal. These video cameras will be addressable via a local network at every intersection, and the proposed research calls for processing these video feeds before they leave this network.

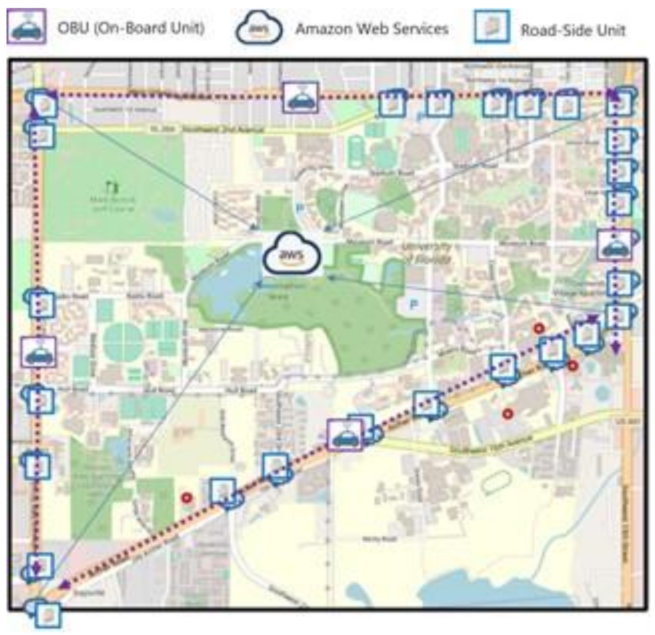


Figure 3-1. Trapezium+ = Trapezium after SPaT project (data sharing and ATSPM; upgraded Linux-based 'ATC' controllers; ATSPM data sharing to University of Florida and third party (TTS, TrafOps, Live Traffic Data, Connected Signals); Siemens DSRC radios with MAP and SPaT broadcast; emergency vehicle pre-emption and vehicle OBUs). The solid arrow lines denote that all collected data will be sent to AWS cloud for data storage.

Research in an ongoing project (BDV31-977-77) has demonstrated the use of edge and cloud computing to process a variety of data sources, including video and signalized intersection data. The research showed how machine learning can be effectively applied to ATSPM data for understanding traffic behavior at a single intersection as well as intersections city-wide. Working with the traffic engineers in the City of Gainesville and Seminole County, the researchers have developed dashboards that allow the user to both visualize and interact with the data for decision making, which was applied in this project.

3.2. Project Objectives

Vehicle loop detectors that have traditionally been deployed at intersections to detect the passage of vehicles have high deployment and maintenance costs and are not always useful for observing the movements of pedestrians and scooters. The use of other modalities, such as video and LIDAR, has great potential to improve accuracy and timeliness in the detection of vehicles, pedestrians, bicyclists, etc. In this project, we augmented the Trapezium infrastructure with additional sensors as follows:

1. Fisheye cameras – Full intersection view, streaming cameras with full count, presence, and queue detectors
2. Infrared Cameras – Nighttime intersection view, streaming cameras
3. LIDAR – For ground-level ground truth verification and pedestrian and bicyclist application.

At a subset of intersections, including the Gainesville, FL, intersections at Gale Lemerand Dr and Stadium Rd and the intersection at NW 13th St W University Ave, this equipment provides a full intersection view, streaming cameras with full count, and presence and queue detectors. The LIDAR provides ground-level truth verification and pedestrian and bicyclist application. The researchers also added gateway computers and communication equipment on these intersections.

We used this edge-based video-stream processing, infrared cameras, and LIDAR at intersections and in public vehicles to convert video data into space-time trajectories of individual vehicles and pedestrians that are transmitted and synthesized on a cloud-based system. Using standard and fisheye cameras mounted at intersections (and integrating loop detectors placed beneath streets), we used state-of-the-art computer vision and machine learning to perform vehicle tracking (localization, tracking, turn estimation) and pedestrian monitoring at intersections. At 20–30 frames per second, high-definition video from a single camera can generate over two gigabytes per hour. Data rates for LIDAR can be larger. This requires GPU-based processing for real-time usage. We used best-of-breed computer vision and machine-learning–based video processing and tracking approaches to generate efficient real-time (and near real-time) representations. Additionally, the following video processing techniques that were developed as part of an NSF-funded project are adapted for this project:

- Multitarget tracking: The objective of multitarget tracking is to use sensor data streams to accurately estimate the position and speed of each object present in the visible surveillance region. Traffic scenes are complex environments with unpredictable behavior, heavy occlusion, and variations in lighting and weather. The project developed efficient and accurate techniques that are robust for such environments.
- Data association and track-to-track association: Two fundamental problems in single-sensor and multisensor multitarget tracking are instances of the multidimensional assignment problem (MDAP) situated within optical flow. The project developed efficient and accurate end-to-end approaches to MDAP.

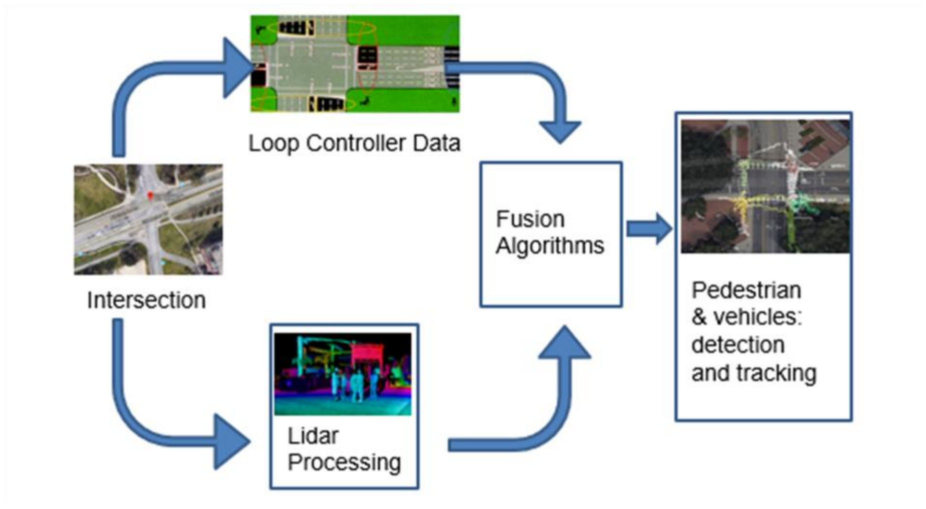


Figure 3-2. LIDAR data collected at intersections is fused with high resolution controller data. The fused data is then used for pedestrian and vehicle detection and tracking by representing these road users as points.

As part of another currently supported FDOT project, the researchers have been working with the City of Gainesville and State of Florida to demonstrate the usefulness of GPU processing of video and ATSPM data for understanding traffic behavior at an intersection. Prototype video processing systems that recognize objects of different types (pedestrian and vehicles) have been developed along with a preliminary trajectory storage and analysis system.

In this task, summarized in Figure 3-2, we extended our work with GPU processing of video and ATSPM data by developing:

1. Video and LIDAR processing techniques and software that can work with a variety of vehicular traffic, pedestrian traffic, changing conditions, and multiple video cameras per intersection
2. Nighttime camera processing techniques and software that are suitable for effectively observing and analyzing nighttime traffic behavior
3. Advanced techniques and software for determination of near-misses, anomalous vehicle trajectories, and incidents
4. Software for near-misses, and other anomalous behavior at the intersection
5. Algorithms for signal optimization that use information from all the video/LIDAR sensors along with information collected from connected and autonomous vehicles.

3.3. Task Overview

Pedestrian counting is challenging, especially in high-volume pedestrian areas (e.g., university environments). Many reported errors occur due to occlusion, especially when counting at high-volume locations. Though manual counts are commonly used, they are expensive and are not feasible at every location. There is also a lack of understanding of pedestrian travel patterns. Although video processing techniques described above can alleviate many of the above concerns, the use of LIDAR has the potential for improving the accuracy of understanding pedestrian/bicyclists behavior at high-volume traffic intersections. In this task, we developed:

1. LIDAR-based techniques to count the number of pedestrians and the patterns of movement they follow while crossing the intersection
2. Techniques to count the number of bicyclists crossing the intersection and the patterns they follow. Using signal timing information, the researchers will quantify the number of pedestrians that remain on the intersection after the light has turned red for their phase.

This work is expected to have a significant impact on understanding traffic behavior at an intersection that will lead to the following:

1. Improved pedestrian and bicyclist safety by examining the conflict points of the vehicle/pedestrian trajectories and anomalous behavior, based on time of day and day of the week.
2. More accurate demand profiles, which can lead to more effective signal timing and a better understanding of the relationships between existing (loop detector) and new technologies (video).
3. Better incident and bottleneck management using real-time information and messaging of incidents and other events to pedestrians and vehicles.

3.4. Literature Review

3.4.1. Point Cloud Processing

Extensive research projects have been undertaken to develop modeling techniques aimed at automatically understanding 3D scenes and objects for numerous applications, such as 3D object classification (Klokov and Lempitsky, 2017; Qi et al., 2017a; Qi et al., 2017b; Li et al., 2018; Wang et al., 2019), 3D object detection (Qi et al., 2018; Shi et al., 2019), 3D semantic labeling (Landrieu and Simonovsky, 2018; Graham et al., 2018; Choy et al., 2019; Su et al., 2018), and 3D instance segmentation (Pham et al., 2019; Yang et al., 2019). Compared to 2D images where data are stored in structured arrays (defined on a regular grid) that explicitly describe neighborhood relations, point clouds have only implicit neighborhood relations because the data are not organized in a pre-defined manner, e.g., point clouds vary on the number and order of points. The unstructured characteristic poses the main challenge of point cloud processing. Most prior work relies on transforming 3D data into regular representations such as voxels (Wu et al., 2015) or 2D grids (Su et al., 2015) for processing. Here, context aggregation can be achieved easily with convolutions at relatively low resolutions due to the expensive computational overhead and memory footprint. To mitigate the issue, the researchers have seen architectures such as OctNet (Riegler et al., 2017) and permutohedral lattice (Su et al., 2018) representation being proposed to conduct efficient memory allocation and computation thus avoiding compromising resolution. Recently, new work has emerged that directly processes raw and irregular point clouds (Qi et al., 2017a; Pham et al., 2019; Qi et al., 2017b; Wang et al., 2019) by applying multilayer perceptrons (MLPs) in a point-wise fashion. To further capture local structures, follow-ups (Qi et al., 2017b; Shen et al., 2018; Wang et al., 2019; Thomas et al., 2019) have defined pseudo-convolutional operators where convolutions are instantiated as continuous kernels, assuming a continuous space for point clouds. However, this incurs an extra cost due to the use of greedy nearest neighbor search and point sampling algorithms for the purpose of hierarchical processing. More recently, sparse tensor-based point cloud processing has been proposed to conduct sparse convolutions only on non-empty locations. Frameworks, such as SparseConvNet (Graham et al., 2018), MinkowskiEngine (Choy et al., 2019), and TorchSparse (Tang et al., 2020), can conduct the sparse convolutions very fast based on their efficient indexing structure.

3.4.2. 2D and 3D Object Detection and Tracking

2D object detection takes one input image and outputs bounding boxes over objects of interests (OIs). The pioneering work includes notable object detectors such as the early work Overfeat (Sermanet et al., 2013), Region-based CNN (RCNN) detector of Girshick et al. (2014), and its follow-ups, e.g., SPPNet (He et al., 2015), Fast RCNN (Girshick, 2015), Faster RCNN (Ren et al., 2015), Deformable-ConvNets (Dai et al., 2017), and MaskRCNN (He et al., 2017). For single shot and fast inference, classical object detectors such as SSD (Liu et al., 2016) and YOLO (Redmon et al., 2016; Redmon and Farhadi, 2017) have been proposed. Recently, the anchor-free object detection has received increasing research attention by looking at high-level image features, predicting center points as well as estimating the scale of objects without requiring a predefined set of anchor boxes. Representative approaches include CornetNet that detects an object bounding box as a pair of keypoints (Law and Deng, 2018), ExtremeNet (Zhou et al., 2019b), CenterNet (Zhou et al., 2019a), FCOS (Tian et al., 2019), and FoveaBox (Kong et al., 2020).

3D object detection aims at estimating three dimensional rotated bounding boxes over images or point clouds. In (Engelcke et al., 2017), the authors use the feature-centric voting to efficiently process point clouds on equally spaced 3D voxels. VoxelNet (Zhou and Tuzel, 2018) uses the PointNet (Qi et al., 2017a) within each voxel to obtain voxel-wise representation, followed by the Region Proposal Network (RPN) (Ren et al., 2015) branch to generate detection. SECOND (Yan et al., 2018) makes a further step to speed up the VoxelNet by using a simplified voxel encoder and a faster implementation of sparse convolutions. SA-SSD (He et al., 2020) improves SECOND by learning more structure-aware features. PIXOR (Yang et al., 2018) instead projects points to a 2D feature map while encoding occupancy and point intensity information. PointPillars (Lang et al., 2019) improves the backbone efficiency by replaying the voxel representation with a novel pillar representation that gathers voxels along the vertical dimension. Inspired by Hough Transform (Duda and Hart, 1972), VoteNet (Qi et al., 2019) clusters the voting to detect objects using point feature sampling and grouping. LIDAR has been used for pedestrian and non-motorized traffic detection in self-driving scenarios, where LIDAR sensors are installed on the vehicles with ego-motion. This includes datasets such as KITTI (Geiger et al., 2012), NuScenes (Caesar et al., 2020), Waymo (Sun et al., 2020), and Argoverse (Chang et al., 2019). Taking the KITTI 3D object detection as one example, the current state-of-the-art 3D pedestrian detector called VPFNet can achieve 48.36%, 54.65%, and 44.98% AP (average precision) on Moderate, Easy, and Hard categories, respectively.

3D object tracking Many 2D tracking algorithms can be adapted to track 3D objects (Bewley et al., 2016; Wojke et al., 2017; Karunasekera et al., 2019; Bergmann et al., 2019). However, to obtain dedicated 3D trackers, the researchers generally extend from 3D Kalman filters (Weng and Kitani, 2019; Chiu et al., 2020) to exploit 3D motion information of a scene. Recently, a much simpler tracker has been proposed in CenterTrack (Cohen et al., 1992) to directly conduct a simultaneous detection and tracking to a pair of images.

3.4.3. Trajectory Clustering

Trajectory clustering is an efficient technique to analyze the trajectories of objects and has been widely used in computer vision and traffic engineering to support applications in pedestrian counting (Antonini and Thiran, 2006), surveillance video analysis (Zhang et al., 2009; Wang et al., 2006; Ge et al., 2012), traffic behavior understanding (Morris and Trivedi, 2011; Hu et al., 2014), and anomaly detection (Piciarelli and Foresti, 2006; Kumar et al., 2017; Piciarelli et al., 2008). Each trajectory of a moving object records the path with space and time information, where each point in the trajectory represents a position in space at a certain instant of time. A clustering algorithm aims at identifying similar patterns of movements and distinguishing undesired behaviors (e.g., outliers).

A suitable measurement must be designed to compute the similarity between different trajectories. Popular measurements include Euclidean distances and Hausdorff distances. The most intuitive Euclidean distance measurement simply sums up the distances between all corresponding pairs of points. However, the limitation is that it requires the number of sample points to be same for all trajectories. For trajectories with different temporal length, it is no longer applicable. This could be handled by Hausdorff distance that measures the distance between two subsets of a metric space (e.g., two trajectories) by finding the greatest distance from a point in one set to the closest point in the other set. Additionally, dynamic time warping (DTW) (Rabiner, 1993) and its variants (Silva and Batista, 2016; Prätzlich et al., 2016) were developed to compute the distance between two trajectories. The longest common subsequence (LCSS) (Vlachos et al., 2002) was proposed to handle noisy trajectories.

A closely relevant approach to the LCSS is the edit distance on real sequence (EDR) (Chen et al., 2005). In general, trajectory clustering algorithms for different objects must be designed differently. For example, a good clustering algorithm designed for vehicles may not be suitable for pedestrians because vehicles are constrained to road networks while pedestrians could walk in a crowded scene with much greater number of trajectories.

3.5. Data Collection and Annotation

This section describes the steps of collecting and annotating LIDAR dataset containing pedestrians at an urban traffic intersection.

3.5.1. LIDAR Data Collection

The LIDAR sequences are recorded at the intersection at Gale Lemerand Dr. and Stadium Rd. in Gainesville, FL, using the Ouster OS-1 16 beam LIDAR (Figure 3-3).

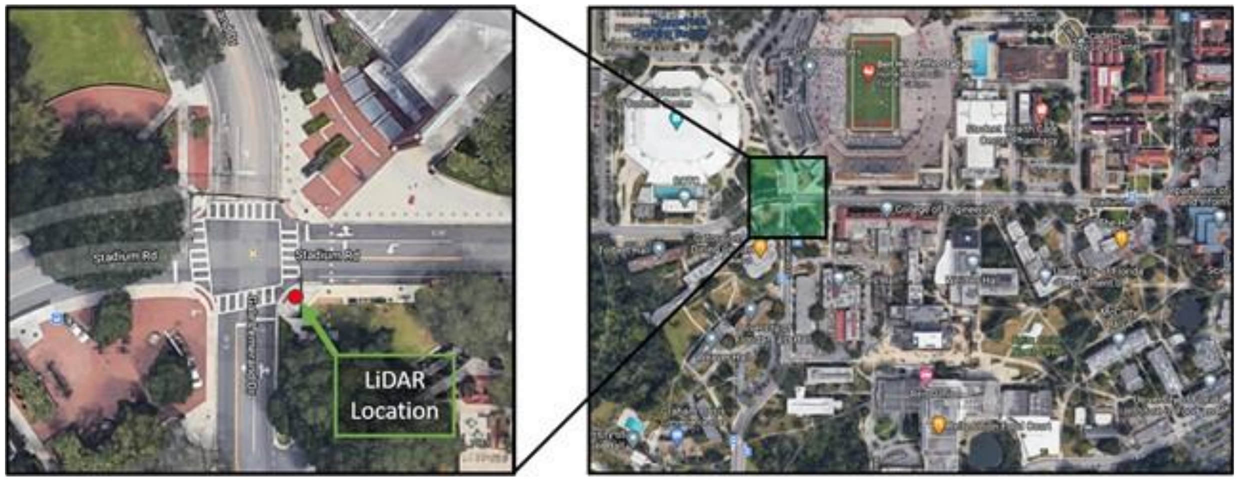


Figure 3-3. LIDAR setup. The LIDAR sensor is installed at an intersection on Gale Lemerand Dr, Gainesville, FL.

The sensor was mounted on a tripod at 2 m from the ground and connected to a rig running on an Intel NUC 6. It is a busy intersection with cars, trucks, motorcycles, bicycles, and pedestrians present. Also, at times there were a few big groups of pedestrians crossing. There are also stationary objects including walls and trees. The OS-1 LIDAR created files containing sequences of raw point cloud data with static background. For bicyclist analysis, the researchers move to a busy intersection at the NW 13th St W University Ave in Gainesville, FL to record LIDAR data on lasting one-hour starting at 15:20 pm, November 19, 2021, using the Ouster OS-1 32 beam LIDAR (Figure 3-4).

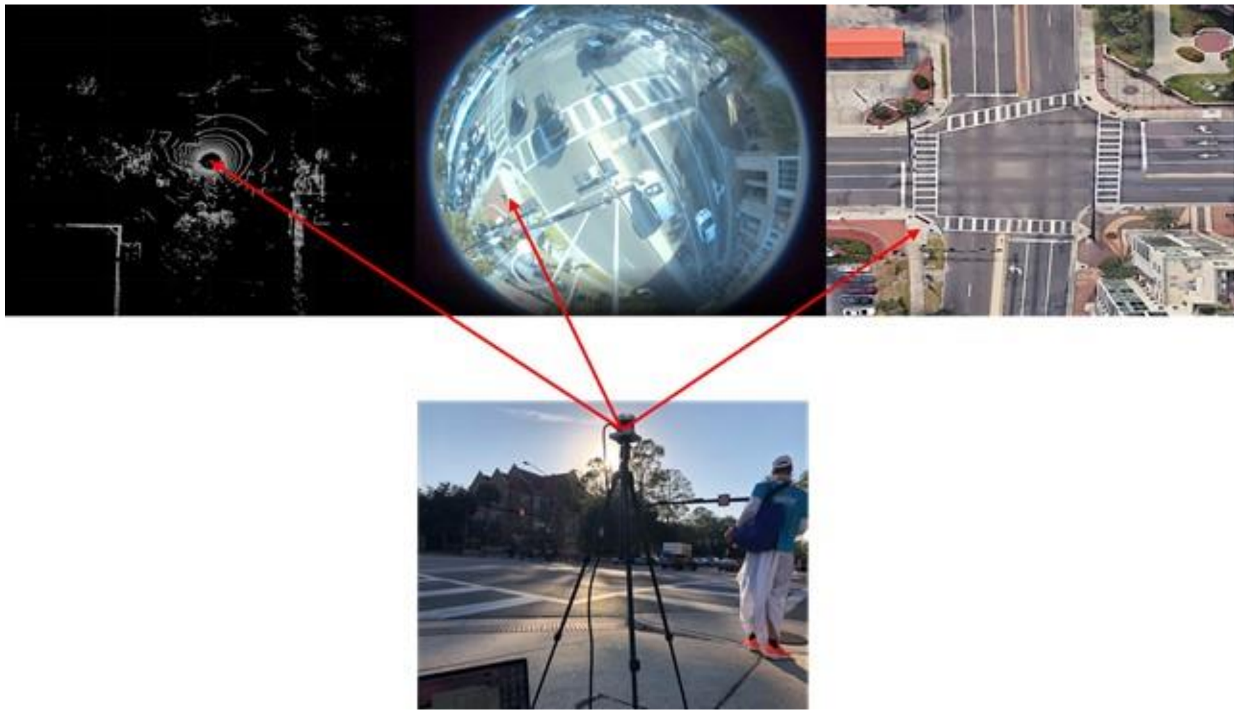


Figure 3-4. Another LIDAR setup. The LIDAR sensor is installed at the busy intersection of W University Ave at 13th St in Gainesville, FL, to collect bicyclist data.

3.5.2. LIDAR Data Annotation

For all recorded sequences, frames are extracted and converted into Point Cloud Data (PCL) files. Each PCL file contains two sections:

- A human-readable header that defines the number, size, dimensionality, and data type of the point cloud.
- In the data section, the binary, non-human-readable data contains points of three dimensional XYZ spatial coordinates and their corresponding reflectance values.

The PCD files are loaded in an efficient annotation tool called SUSTechPOINTS (Li et al., 2020). It supports various useful features such as nine degrees of freedom (DOF) box editing, editing on perspective view and projective views, semi-auto box annotation, focus mode to hide background to check details easily, interactive box fitting, and batch editing (Figures 3-5, 3-6, and 3-7). The workers carefully go through each frame to draw 3D bounding boxes over moving objects and label them into road user classes such as pedestrians, riders, buses, and cars. Also, the tool provides one unique ID for each instance object for tracking it across all frames where it appears, which essentially provides the ground-truth trajectory for the given object. The work only considers a region defined by [−51.2, 51.2] × [−51.2, 51.2] × [3, 5] for each frame to study the behaviors of road users.

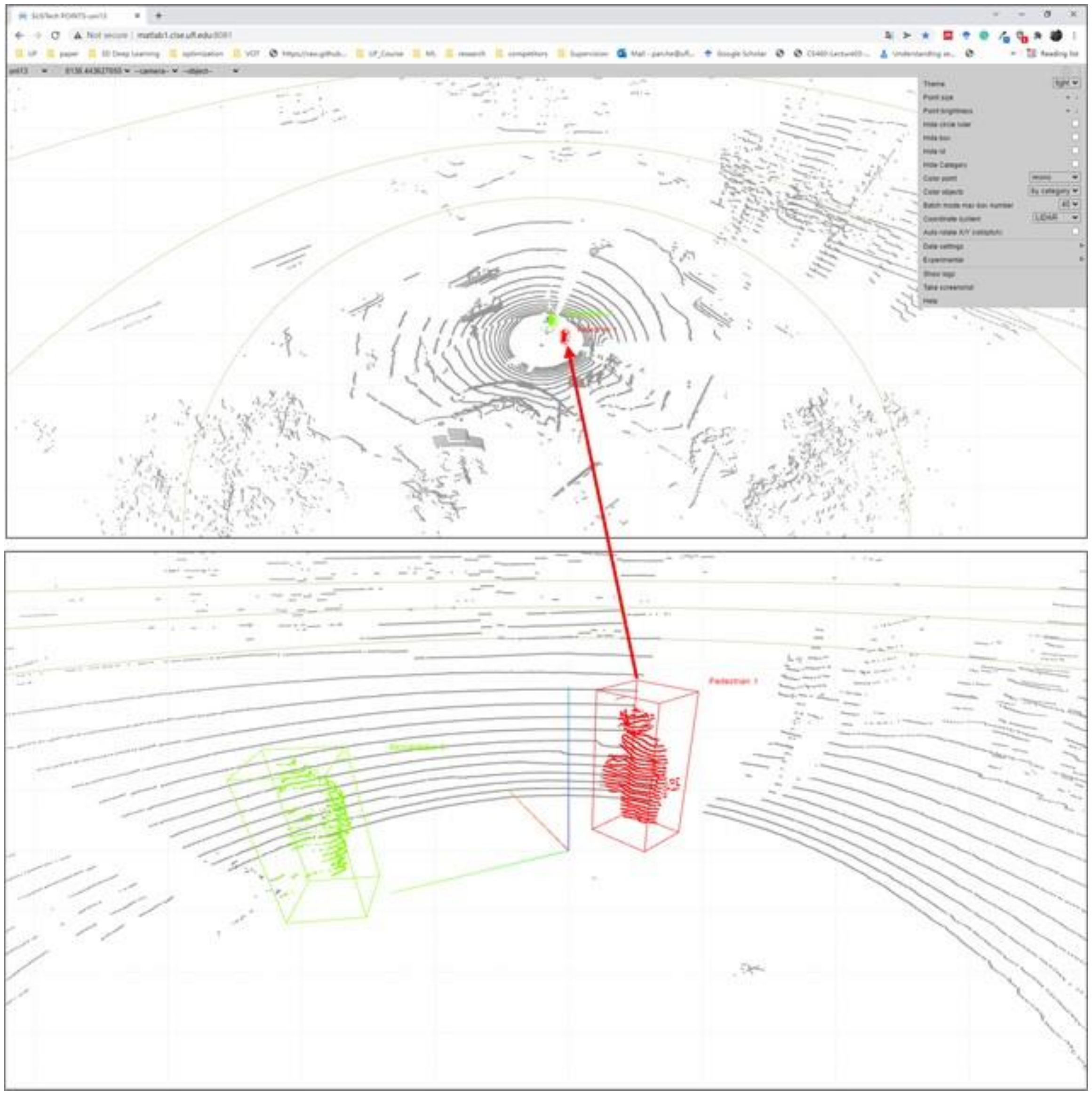


Figure 3-5. Example annotated data using the LIDAR annotation tool. Each object is annotated with its class type and instance id, and draw a 3D bounding box over it. The instance ID is consistent across multiple frames to track a certain object.

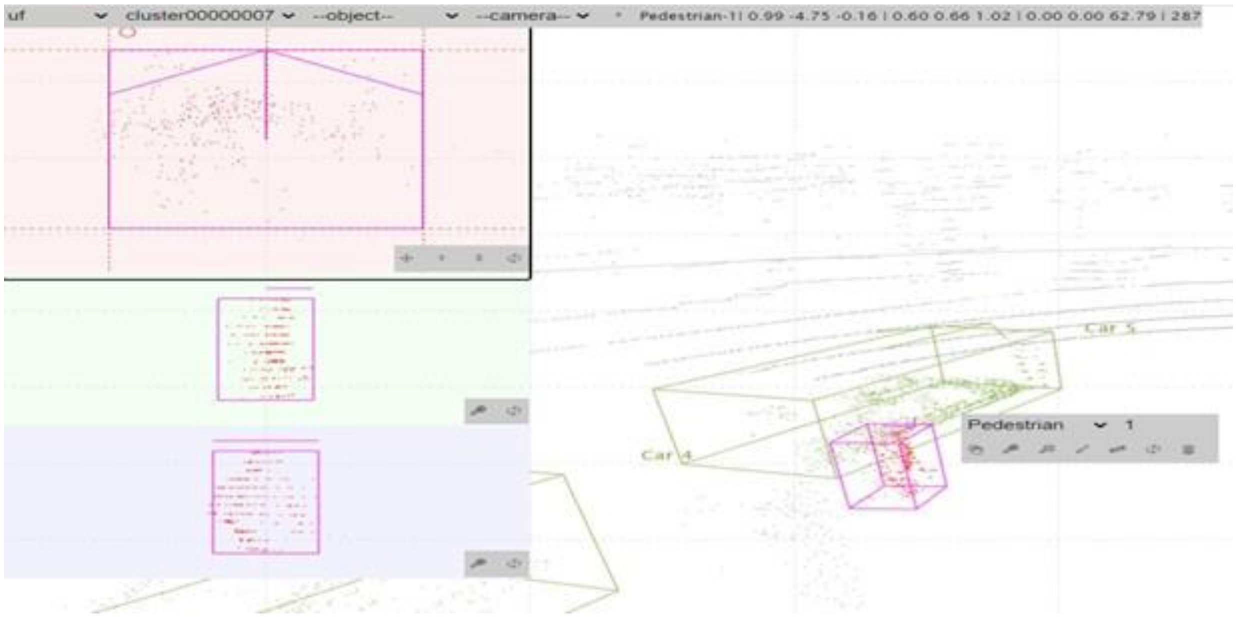

uf
cluster00000007
--object--
--camera--
Pedestrian-1| 0.99 -4.75 -0.16 | 0.60 0.66 1.02 | 0.00 0.00 62.79 | 287
Car 5
Car 4
Pedestrian
1

Figure 3-6. Multiple views for an object. The left panel consists of a top view (top), a side view (middle) and a front view (bottom) to help the user annotate in 2D. The right-hand figure demonstrates the 3D view of a point cloud frame where the user can navigate and place annotations.

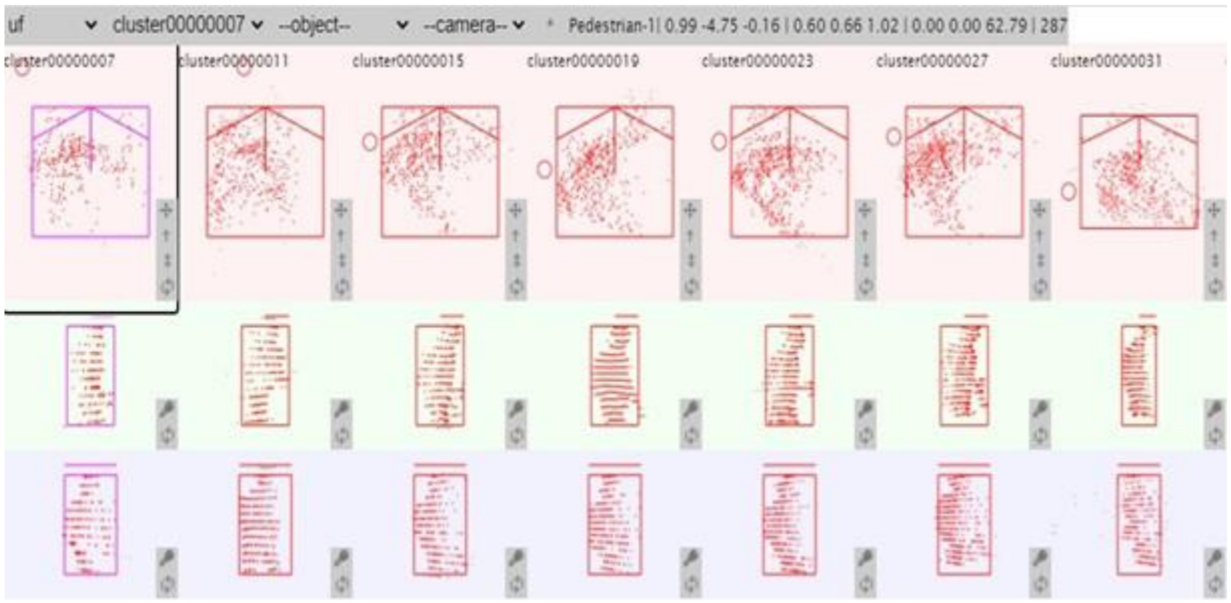


Figure 3-7. The batch-edit feature of the annotation tool. Given one annotated object at the current frame, the batch-edit can propagate or interpolate the annotated 3D bounding box to its subsequent frames, which could improve the annotation speed. For example, a pedestrian is stopping and waiting at the intersection. In a relative long time window, it remains still and could be quickly annotated by a direct 'copy&paste' of the previous annotation.

3.6. Methods

In this section, the details of processing the LIDAR for pedestrian analysis are discussed. And a necessary background is presented before introducing the actual method.

3.6.1. Preliminary

Denote by P = $\{(x, y, z, r)\}_i$ an orderless set of point clouds with (x, y, z) representing location for each point and r being the reflectance value. The 3D object detection task will try to predict a set of 3D bounding boxes B = $\{b_k\}$ where each $b_k = (u, v, d, w, l, h, \alpha)$ contains the centroid location (u, v, d), the size (w, l, h), and the yaw angle α. The sensor is located at the origin (0, 0, 0) of the 3D coordinate system with a yaw angle $\alpha = 0$.

When the number of points is huge, most modern 3D object detectors will use the representation rather than the raw point representation. This involves a preprocessing step to transform the raw point cloud into a set of voxels (or Pointpillar, or bins) and extract features from all points within each voxel via a PointNet-like network (Qi et al., 2017a). The detectors pool these features to obtain a compact representation and feed them into a 3D backbone network (usually it is a sparse convolutional network). These encoded features will be further flattened into a planar map-view and passed through a stack of 2D convolutional backbone. In the end, they obtain an output feature $F = R^{W \times L \times C}$ where W, L and C are the height, width, and channel size of the feature. The researchers then follow the standard 2D object detection approaches such as RPN (Ren et al., 2015) and CenterPoint (Yin et al., 2020) and add the detection branch to produce candidate detections from the overhead feature map. This project follows CenterPoint (Yin et al., 2020) to adapt off-the-shelf 3D encoders, namely VoxelNet (Yan et al., 2018, Yang et al., 2018), PointPillars (Lang et al., 2019), and CenterNet (Zhou et al., 2019a).

VoxelNet quantizes the LIDAR points into a set of voxels and discards points out of the quantization range. Within each voxel, a PointNet is applied to extract 3D point features, followed by a per-voxel max- or mean-pooling to flatten them into a feature vector of a fixed size. Sparse convolutions (Graham et al., 2018, Choy et al., 2019) are then applied to obtain 3D voxel features. They are further collapsed into a map-view representation by gathering all voxels along the height dimension following PointPillars (Lang et al., 2019). A 2D CNN further encodes the map-view representation and produces the feature F.

CenterNet treats object detection as keypoint estimation where a heatmap $Y \in \{[0, 1]\}^{w \times h \times k}$ is predicted for K classes. Then each local maximum in the heatmap is considered as the centroid location of one candidate object. To obtain the 2D size estimation of the detected object, CenterNet further regresses a size map $S \in R^{w \times h \times 2}$ to describe height and width of the corresponding bounding boxes. CenterNet is a pure fully convolutional neural network architecture with a dense prediction branch. The training and inference details of CenterNet can be found in the original paper (Zhou et al., 2019a).

3.6.2. CenterNet-based Tracking

Inspired by CenterNet (Zhou et al., 2019a) and CenterPoint (Yin et al., 2020), a center-based detection head is utilized to detect 3D objects from LIDAR point clouds. The design is similar to 2D CenterNet. The output of the 2D backbone is used for producing multiple branches: 1. The heatmap branch to predict centroids of K classes; 2. The dense object size prediction branch to regress the width, length, and height (w, l, h) of a 3D bounding box attached to one object. Additionally, to track objects across frames, anther tracking branch is added to predict a three-dimensional motion offset o = (ox, oy, oz) and another box rotation estimation branch to predict yaw-angle-related value (cos(α), sin(α)).

The Hungarian algorithm is a combinatorial optimization algorithm solving the assignment problem. It can be used in tracking to associate new detection results to existing tracked objects, each object will use the predicted motion offset from the tracking branch to obtain the new location in next frame. Then an assignment matrix is constructed as the Euclidean distance between each new detection and all predicted bounding boxes of the existing tracked objects. The optimal solution is obtained using the Hungarian algorithm. A minimal distance threshold is added to constraint the matching processing. A matched track with a Euclidean distance lower than the minimal distance threshold will be considered as a new track. After the Hungarian matching, all unmatched tracks will be stored if its age (the total number of frames since first occurrence) doesn't exceed the max age (the maximum number of consecutive misses before the track state is set to 'Deleted'). This is a reasonable implementation considering the situation where a tracked object is temporarily occluded in a short period. Should it reappear in the scene, its tracking is resumed with the original identity. However, we will not output these tracks in the current frame. The max age will be also used to delete a tacked object when the object enters and leaves the scene. There are other heuristic settings such as the detection threshold and a minimal object size.

3.6.3. Trajectory Clustering

The project follows (Atev et al., 2010) for clustering tracked objects. For a given set of trajectories, it groups trajectories such that trajectories in the same group are more similar to each other than to those in other groups. The pipeline consists of several major components:

1. Compute the similarity scores between all pair trajectories with a appropriate metric.
2. A cluster algorithm on the computed similarity matrix to generate trajectory clusters.
3. For a new trajectory, it is forwarded to the trained cluster model and obtain the track-to-cluster assignment.

To obtain the similarity matrix, the Hausdroff distance is utilized as the similarity metric. The Hausdorff distance is defined as the maximum distance of a set to the nearest point in the other set (Rote, 1991). More formally, the Hausdorff distance from set A to set B is defined as

$$h(A,B) = \max_{a \in A} \left\{ \min_{b \in B} \{ d(a, b) \} \right\} \qquad 3.1$$

where *a* and *b* are points of the two sets *A* and *B,* respectively. *d(a, b)* is a metric between point a and b, which is generally a Euclidean distance.

A naive way would be utilizing the SciPy library with a single thread CPU, which could take hours to obtain the matrix if the number of trajectories is huge. Instead, the project speeds up the computation by using the state-of-the-art GPU-based cuSpatial library. The library is originally designed for accelerating geospatial and spatiotemporal processing. Due to the efficient parallel implementation, the running time could be reduced to minutes.

For the clustering algorithm, two popular approaches are considered, namely the agglomerative clustering (Nielsen, 2016) and density-based spatial clustering of applications with noise (DBSCAN) (Ester et al., 1996). The former is a classic bottom-up hierarchical clustering method that starts with all observation with their own clusters, and then merges pairs of clusters as one moves up the hierarchy. The latter one is a density-based clustering non-parametric algorithm that groups points that are packed together and marks outlier points lying alone in low-density regions. All implementations are written in Python with the installation of all necessary libraries such as Pytorch, cuSpatial. GPU accelerators are used to speed up the training and inference as well.

3.7. Results and Analysis

In this section, preliminary results obtained from our current development are presented, followed by a further discussion on how to better visualize the point cloud with the developed annotation tool, how well the 3D object detection and tracking model performs, and the clustering results for trajectories.

3.7.1. Bird-View Visualization

To better visualize the LIDAR point clouds, the project overlays the google satellite image and the LIDAR data at the intersection and projects it into a 2D bird-view map (Figure 3-8). To do so, it first projects the 3D point cloud to the 2D map by discarding the height dimension. Then a worker manually selects several keypoints that are matched between the two images to compute a homography matrix. This matrix is used to project the LIDAR image to the reference satellite image. Notice that the LIDAR is installed on the road near the ground. Therefore, when a pedestrian is approaching to the sensor location, the pedestrian will occlude a large portion of scene thus creating a empty frustum-like space behind it.

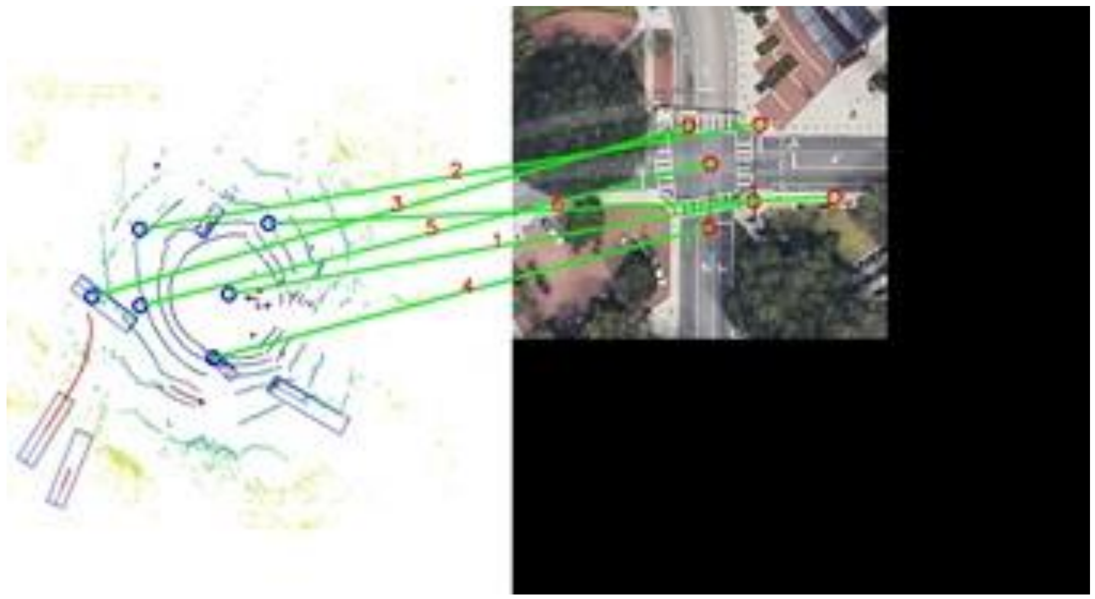


Figure 3-8. The developed homography annotation tool for overlapping LIDAR and Google satellite images. Matching keypoints between the pair images are selected and used to further compute the homography matrix. (a) manual keypoint matching; (b) the overlayed image.

3.7.2 Pedestrian Detection

Evaluation Metrics 3D bounding boxes are manually annotated for all pedestrians, which compare against the detected boxes. Mainly, the precision, or positive predictive value, is considered, which is defined:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \qquad 3.2$$

It identifies approximately the ratio of correctly detected pedestrians to the total number of objects perceived by the detector to be a pedestrian. A precision close to 1 signifies that few objects (e.g., cars, trucks, signposts) were confused for a pedestrian. The recall measures the fraction of pedestrians correctly identified out of all possible pedestrians in the dataset. That is, it answers the question of "how many do we miss?". Formally, it is defined as:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \qquad 3.3$$

A recall close to 1 signifies that a method does not miss any pedestrians. Finally, the F1-score is the harmonic mean of precision and recall and provides a holistic performance measure. An F1-score of 1 would be an ideal classifier.

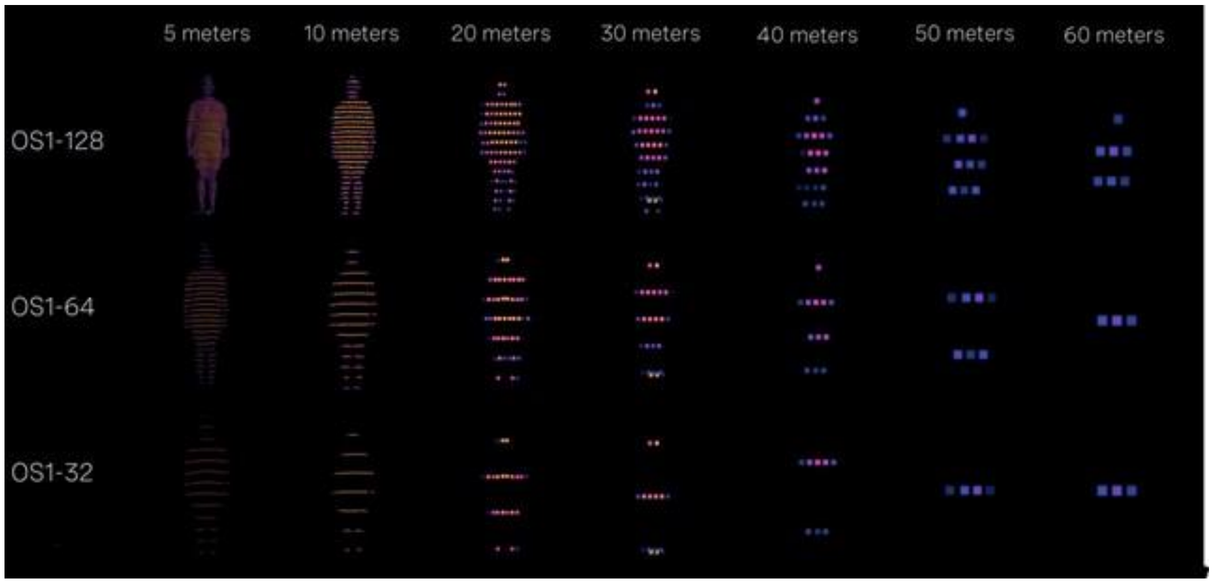


Figure 3-9. Visualization of pedestrians under different sensor configurations

Challenges – Figures 3-9 and 3-10 demonstrate challenges of pedestrian detection using LIDAR point cloud. When objects move away from the LIDAR sensor, the LIDAR sensor has less chance to hit on the surface of pedestrians thus creating fewer corresponding point clouds. Losing the resolution will cause challenges on detecting and identifying pedestrian objects. Moreover, when a pedestrian approaches to our LIDAR sensor deployed at a near-ground (0.5 m above the ground plane) height, it tends to create a heavy occlusion that all scenes and objects behind the pedestrian will disappear.

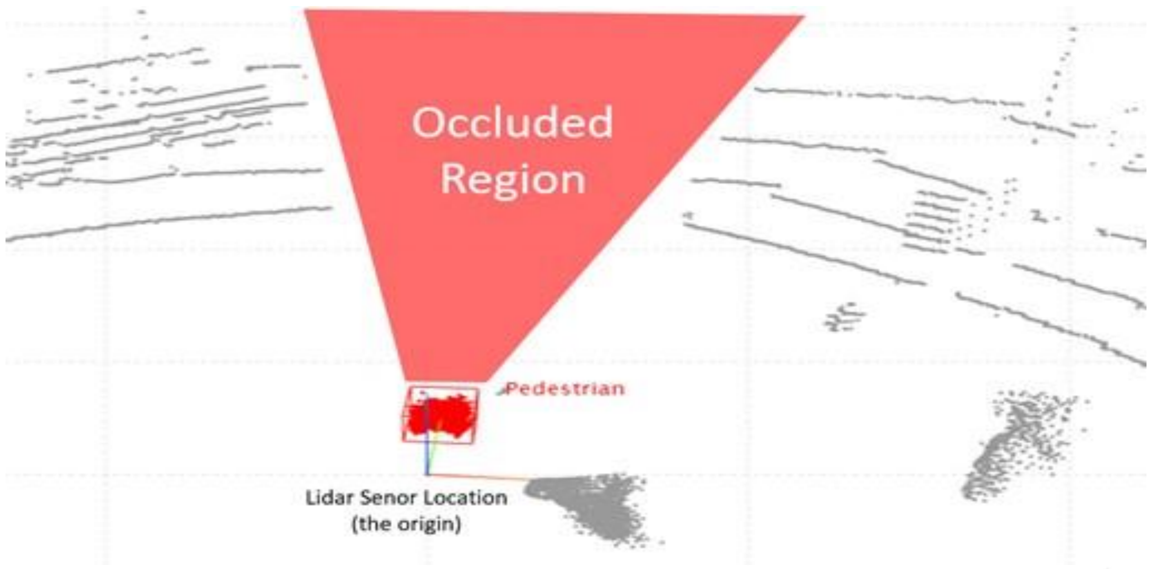


Figure 3-10. The occlusion issue created by a pedestrian approaching to the LIDAR sensor.

Key Results – Tables 3-1 and 3-2 provide the results of the model on our collected test data. The project adds double-flip testing where the input point cloud is forwarded to the model using all four combinations of horizontal and vertical flipping. These results will be averaged to generate an ensembled output in Table 3-2.

Table 3-1. Model performance with PointPillar backbone

Range	True Positive	False Positive	False Negative	Recall (%)	Precision (%)	F1-score (%)
10 meters	997	870	496	66.78	53.40	59.35
15 meters	1300	1251	761	63.08	50.96	56.37
20 meters	1362	2102	831	62.11	39.32	48.15
30 meters	1373	3431	866	61.32	28.58	38.99

Table 3-2. Model performance with VoxelNet backbone on pedestrian detection.

Range	True Positive	False Positive	False Negative	Recall (%)	Precision (%)	F1-score (%)
10 meters	1224	571	269	81.98	68.19	74.45
15 meters	1673	882	388	81.17	65.48	72.49
20 meters	1759	2113	434	80.21	45.43	58.00
30 meters	1775	3202	464	79.28	35.66	49.20

Model with PointPillar Backbone – The project achieved test set performance of a 59.35% F1-score when considering the area within a range of 10 meters from the LIDAR sensor.

The corresponding precision and recall are 53.40% and 66.78%, respectively. It is observed that when the detection range is increasing from 10 meters to 30 meters, the recall remains relatively stable while the precision drops significantly, which is due to the loss of resolution on far regions in which intersection poles might be detected and classified as pedestrians.

Model with VoxelNet Backbone and Flip-testing – The project further employs a stronger VoxelNet backbone architecture to allow for a higher voxel resolution, which enables us to capture finer details of objects. This is critical for pedestrians because they are much smaller compared to larger objects (e.g., cars, buses). Finer resolution allows to 'zoom-in' for better detection performance on pedestrians. It also adds double-flip testing such that the model feeds the input point cloud multiple times with a combination of horizontal and vertical flipping. These results will be averaged to generate an ensembled output.

These techniques significantly improves the performance. Compared to PointPillar-based model, the model obtained the F-1 score 74.45%, significantly improves the original results of 59.35%. The recall has reached to 81.98%, which is reasonable for reality usage. When deploying LIDAR with a higher number of beam, e.g., 64 or 128-beam, the models are expected to achieve a better accuracy. As shown in Figure 3-9, higher number of beams capture pedestrians in higher resolution, capturing more details. Higher-quality data collection could lead to better accuracy. It is also promising that further improvement can be made from the reliability of the detector by adding more training examples and explore advanced domain adaptation techniques such as (Wang et al., 2020).

3.7.2. Bicyclist Detection

Similarly, the developed models are used to evaluate the bicyclist detection as well (Table 3-3 and Figure 3-11). Compared to pedestrians, bicyclists appear significantly less frequently at intersections and is more challenging compared to pedestrian detection, which is shown in Table 5. The detection performance is also similar to the 3D detection on nuScenes test set of the state-of-the-art CenterPoint (Yin et al., 2020) where the bicycle detection achieves a mAP of 28.7%. As our LIDAR sensor is installed on the road-side, bicyclists suffer more from occlusions by other pedestrians. For example, pedestrians tend to form a group when walking across an intersection, which is more likely to occlude all objects behind them. The results indicate that bicyclist detection remains a challenging problem.

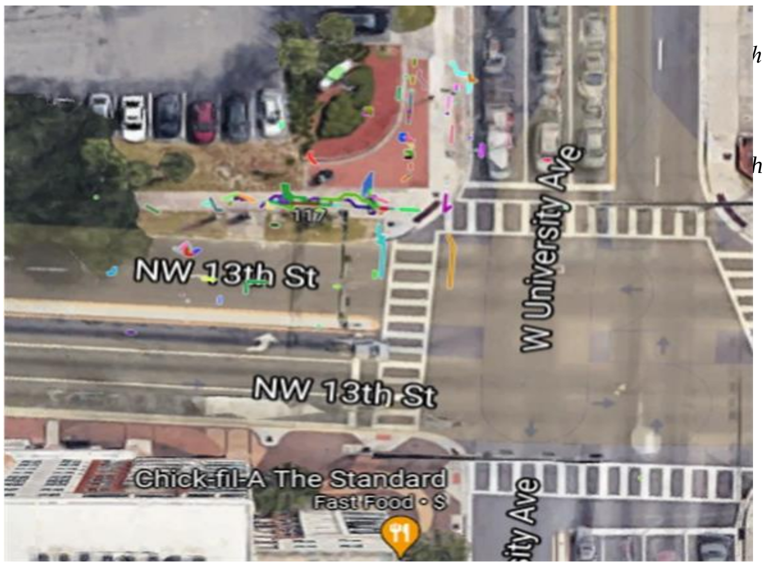
Table 3-3. Model performance with VoxelNet backbone on bicyclist detection.

Range	True Positive	False Positive	False Negative	Recall (%)	Precision (%)	F1-score (%)
5 meters	258	465	878	22.71	35.68	27.75
10 meters	259	658	1147	18.4	28.24	22.30
15 meters	259	862	1240	17.27	23.10	19.77
20 meters	259	1001	1304	16.57	20.55	18.34
30 meters	259	1229	1356	16.03	17.40	16.69

3.7.3. Trajectory Clustering

Once obtaining all detected trajectories, the project will analyze the patterns of these trajectories, via the trajectory technique previously introduced in Section 3.6.3. The main goal is to find routes of pedestrians and potentially detect any abnormal pedestrian trajectories by treating them as the outliers.

Key Results – Figures 3-12 and 3-13 provide the clustering results of the generated trajectories. Additionally, the project visualizes results of car and bus instances (Figures 3-14 and 3-15). It reasonably identifies key routes for pedestrians at this intersection.



h

h

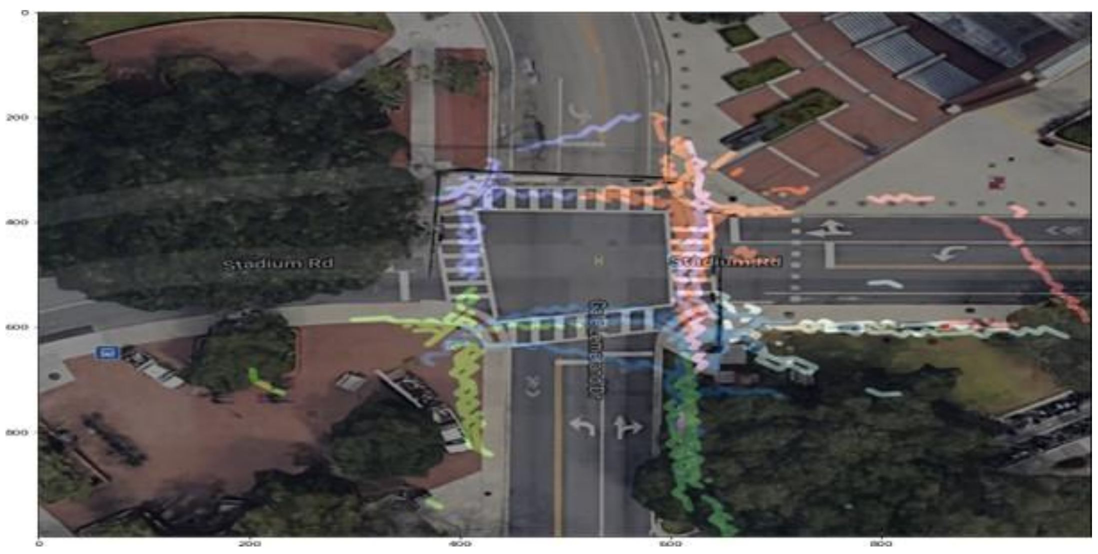


Figure 3-12. Trajectory clustering for pedestrians, generated by Agglomerative Clustering

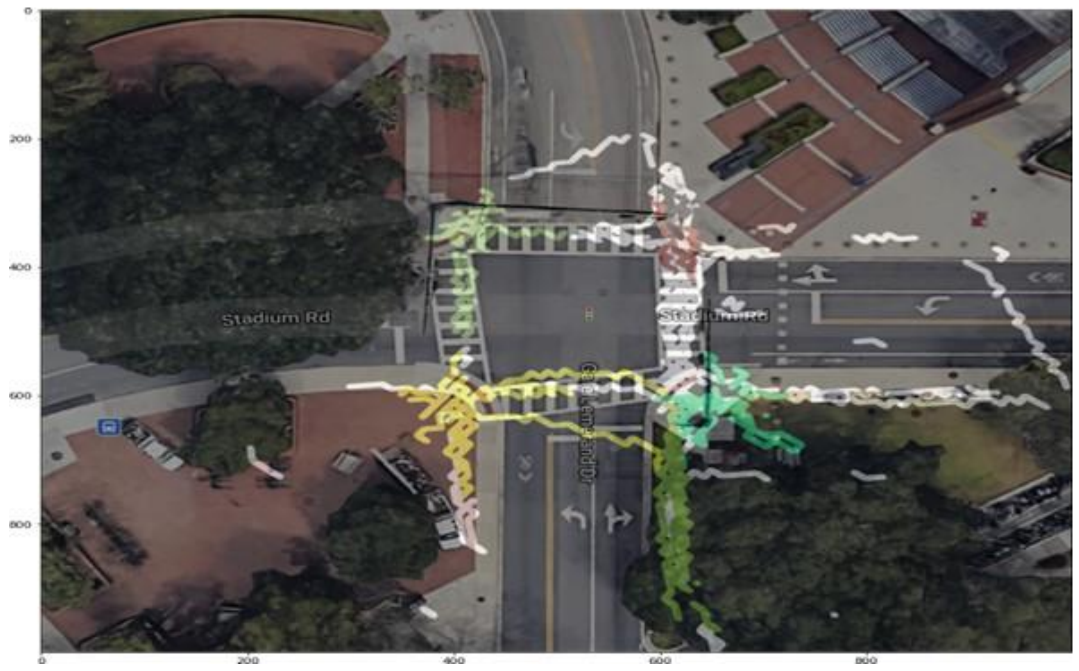


Figure 3-13. Trajectory clustering for pedestrians, generated by HDBSCAN Clustering

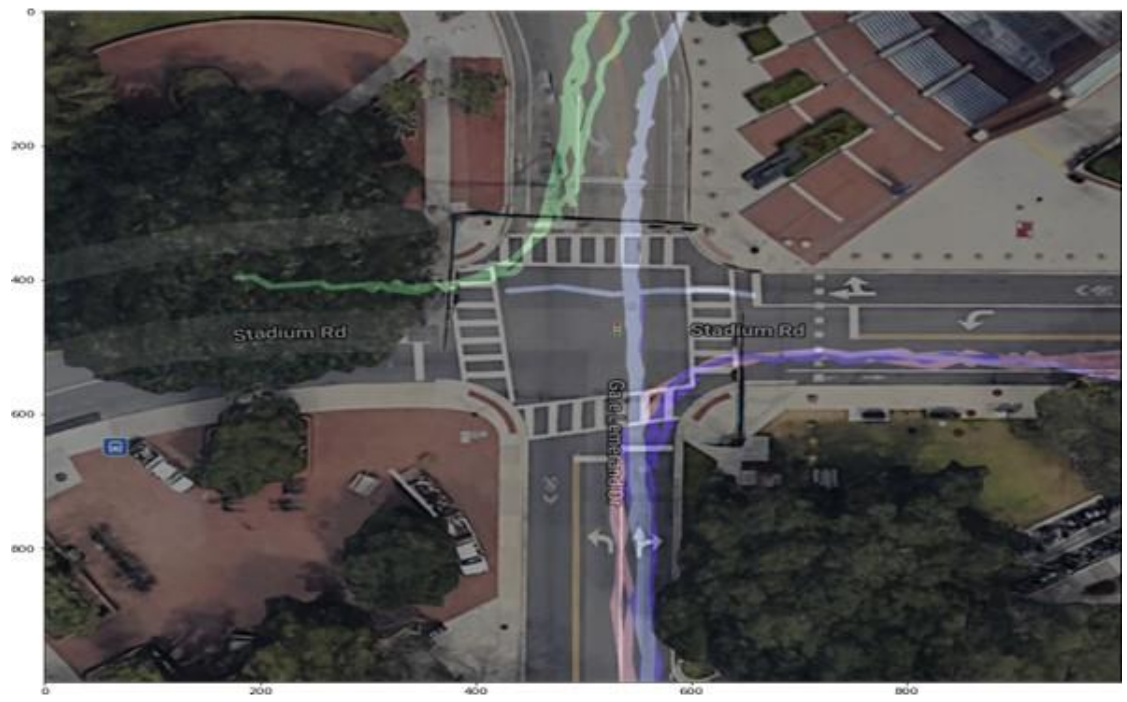


Figure 3-14. Trajectory clustering for buses, generated by Agglomerative Clustering

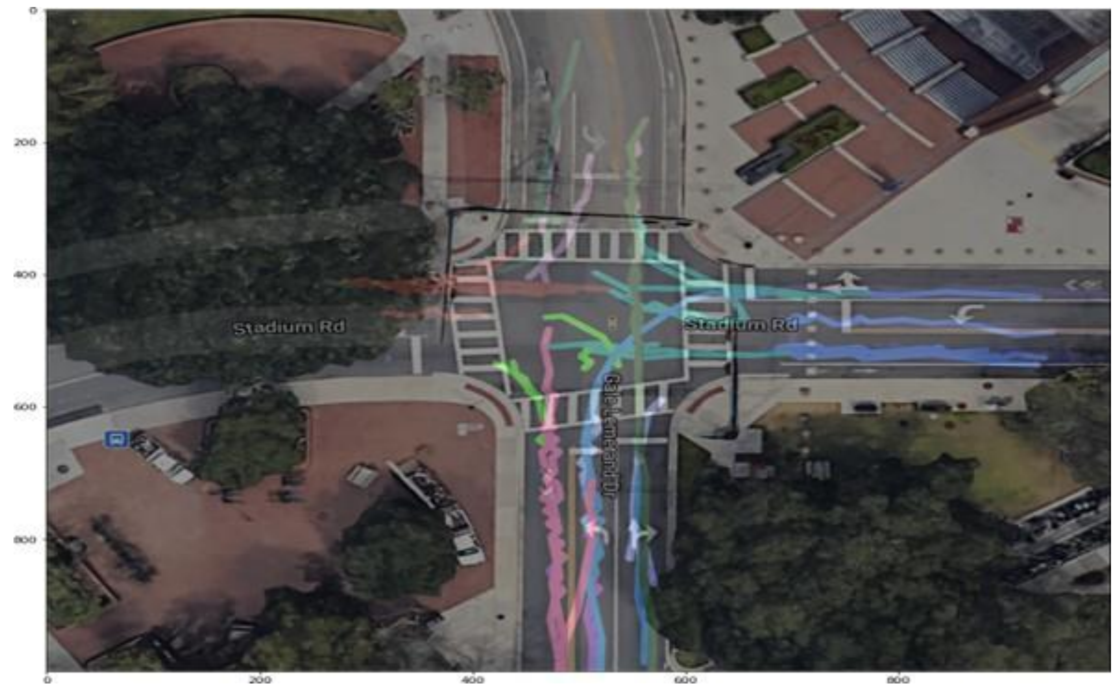


Figure 3-15. Trajectory clustering for cars, generated by Agglomerative Clustering

3.8. Conclusions

The report has presented a system for detection, tracking, and clustering of pedestrians "in-the-wild" that can be used to study the patterns of movement they follow at the intersection.

The approaches rely on recent advances in deep learning for 3D object detection, track-ing and classification as well as conventional clustering methods such as Agglomerative Clustering and DBSCAN. The project developed a homography annotation tool to overlap google satellite image and the collected LIDAR data, achieving a bird-view visualization for LIDAR data. The project developed a pedestrian detection and tracking approach that combines state-of-the-art 3D object detectors with conventional data-association techniques for efficiently tracking pedestrians. From these, it has obtained trajectories for road-users such as pedestrians, buses, and cars. For a given set of trajectories, it grouped trajectories such that trajectories in the same group are more similar to each other than to those in other groups. To achieve this, the project has identified and utilized appropriate metrics to measure similarity between all pair trajectories, and further developed clustering algorithms to obtain the track-to-cluster assignments, which reasonably identifies key routes for pedestrians at one intersection.

All the results obtained have been presented in Section 7. These results show that the developed schema for pedestrian detection has achieved > 80% high recall and > 70% F1-score for detecting pedestrians on OS1-16 LIDAR sensors at various ranges and has potential for wider deployment. The clustered trajectories from our developed algorithms provide insights on the moving patterns at the intersection. Compared to video cameras, LIDAR is generally more challenging to get installed and maintained at traffic intersections. However, LIDAR sensors have their unique advantages of providing precise location information of objects of interest. From this study, it suggests that a mounted location that is sufficiently high is necessary to reduce the occlusion created by objects approaching to the sensor, which could further improve the performance. As LIDAR sensors become more accurate and cheaper, it is promising to be used as a reliable option at the edge in service of intelligent traffic intersection systems. The project will start looking at signal timing once the data is available and cleaned

Chapter 4 – Develop and Test Fusion Algorithms for ATSPM, Video-based and LIDAR-based Data

4.1. Introduction

Video- and LIDAR-based trajectory data generally lack information about signal timing, which can be derived from ATSPM. In this task, we developed and implemented techniques for merging information from video, LIDAR, and ATSPM data streams. Clock synchronization between multiple data sources is a challenging problem and was addressed. The outcome of this task was a system for estimating turn movement counts and continuous-time demand profiles organized by vehicle type (such as car, pedestrian, bicyclist, scooter, etc.). Trajectories obtained from video and LIDAR will be decomposed by phase and synchronized to the intersection signal timing data.

The system will use video processing to obtain trajectories generated by motor vehicles (e.g., cars, trucks, and motorcycles) and LIDAR to obtain trajectories for pedestrians and bicyclists. LIDAR sensors generate precise 3D point clouds that are well-suited for detecting foot and bicycle traffic. By contrast, fisheye video contains heavy distortion that negatively impacts its ability to detect these classes of road users. We employed a late fusion strategy to combine the multi-sensor trajectory data. Late fusion was applied as a post-processing step after each sensor has produced its own set of sensor-centric trajectories. The system also ingests ATSPM data that contain signal timing information recorded at the decisecond level. However, ATSPM data must be synchronized with the video and LIDAR data before being used for analysis. The system uses image processing techniques to detect signal change events in the video feed. These events are then used as a reference point for temporal synchronization.

The solutions for processing video and LIDAR data are based on prototype systems already developed as part of the currently funded FDOT data analytics project. Video processing involves first removing distortion from fisheye video streams and then applying state-of-the-art video tracking algorithms. LIDAR processing involves applying state-of-the-art 3D tracking algorithms to dynamic point cloud sequences. The sets of trajectories from both sensors are mapped to a shared 2D birds-eye-view global coordinate system where they are joined. The processed trajectory data is stored in a database and can be visualized through a Web user-interface for end users.

4.2. Literature Review

In this section, we review recent work on multi-sensor road user analysis. Past research has explored the advantages of using an array of homogeneous or heterogeneous sensors for observing traffic. For applications that require detecting and tracking vehicles at traffic intersections, fusing video and radar streams has been shown to be an effective strategy (Roy et al., 2011; Emami et al., 2021). A system for parking lot and intersection monitoring based on multi-sensor multi-object tracking using a group of networked cameras was recently introduced (Nikodem et al., 2020). Multicamera systems have also been explored for traffic volume detection and prediction (Peppa et al., 2021). A multi-LIDAR system has been developed for monitoring objects within an intersection with reasonably high accuracy (Zhao et al., 2011). However, LIDAR sensors will miss objects when they are hidden behind other objects. As a result, many sensors are needed to be installed at various locations around the intersection, which increases the installation and maintenance costs of this system. ATSPM logs were used for the analysis of pedestrian behavior at a city-wide scale (Singleton and Runa, 2021). The multi-sensor system we develop is unique in that we used LIDAR, video, and ATSPM to develop a comprehensive system for analyzing the trajectories of pedestrians, bicyclists, and vehicles around a signalized traffic intersection. Sensor data is combined with a late fusion strategy in our system. The alternative strategy, early fusion, seeks to use each sensor's raw, noisy object detections and used exceedingly complex algorithms to associate and merge detections prior to obtaining a set of trajectories for all road users. Late fusion is preferable for our setting since the sensors are heterogeneous (of different modality) and provide information about different classes of objects.

4.3. Methodology

In this section, we provide a detailed description of each component of the developed system.

4.3.1. Fisheye Video Processing

The fisheye video processing pipeline was initially presented in detail in Task 1 of this FDOT project (as shown in Figure 4-1). Here, we summarize the pipeline for completeness. We aim at detecting and tracking vehicles in fisheye video and approximately estimating their locations in the real-world on an overhead satellite map of an intersection. This involves first removing fisheye distortion from video frames and then detecting and tracking objects. Distortion corrected trajectories and speed information of road objects are later utilized to detect near-misses in spatial and temporal domains, respectively. We specify four categories of vehicles of interest: motorbikes, cars, buses, and trucks. The overhead satellite maps of intersections are cropped from Google Earth®.

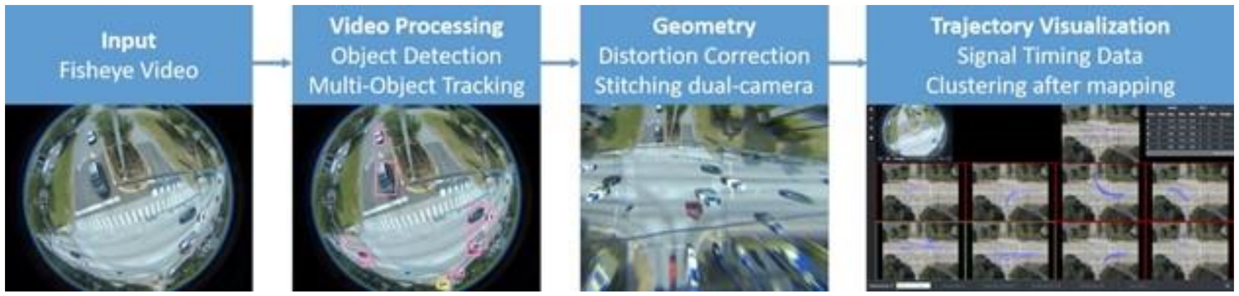


Figure 4-1. An overview of the fisheye video processing pipeline consisting of object detection and multi-object tracking for trajectory generation, followed by post processing and fusion with SPaT data and a final visualization.

We first apply camera calibration methods on a fisheye background image (with no road objects) to make an initial correction. A thin-plate spline projection is then estimated that maps points from a fisheye image to a reference overhead satellite map image. We train an object detector using deep learning techniques (You Only Look Once (YOLO) [Redmon et al., 2016]) and design a vehicle re-identification model with deep cosine metric learning to handle occlusion problems. We integrate these two models into a multi-object tracking pipeline. Given fisheye videos as the input, it supports real-time object detection and multiple object tracking. Using the estimated transformations for the fisheye distortion and reference image mapping, we correct and scale road object trajectories and speed information from the perspective of the overhead satellite map with learned deep features. As the complexity of coordinates transfer is O(1), it allows us to process data both online and offline.

4.3.2. LIDAR Processing

The LIDAR processing pipeline was initially presented in detail in Task 3 of this FDOT project and is summarized again here for completeness. Inspired by CenterNet (Zhou et al., 2019) and CenterPoint (Yin et al., 2020), an object-center-based detection head is utilized to detect 3D objects from LIDAR point clouds. The considered classes are pedestrians and bicyclists. The output of a 2D backbone that processes point clouds from a birds-eye-view is used as input for multiple prediction branches. First, we have a branch to predict centroids of objects as a heatmap. Second, an object size prediction branch is used to regress the width, length, and height (w, l, h) of a 3D bounding box for each object. A third tracking branch predicts a three-dimensional motion offset $o = (o_x, o_y, o_z)$ for each centroid. A fourth rotation estimation branch predicts yaw-angle rotation for each bounding box.

To associate new 3D detections to existing tracked objects we use the Hungarian algorithm. The Hungarian algorithm is a combinatorial optimization algorithm for solving assignment problems. For each object, we use the predicted motion offset from the tracking branch to obtain the new location in next frame. Then an assignment matrix is constructed as the Euclidean distance between each new detection and all predicted bounding boxes of the existing tracked objects. The optimal solution is obtained using the Hungarian algorithm. A minimal distance threshold is added to constraint the matching processing. A matched track with a score lower than the minimal distance threshold will be considered as a new track. After the Hungarian matching, all unmatched tracks will be stored if its age doesn't exceed the max age. This is a reasonable implementation considering the situation where a tracked object is temporarily occluded in a short period. Should it reappear in the scene, its tracking is resumed with the original identity. However, we will not output these tracks in the current frame. The max age will be also used to delete a tacked object when the object enters and leaves the scene. There are other heuristic parameters such as the detection threshold and a minimal object size that affect the performance.

4.3.3. Video and LIDAR Data Fusion

We used the late fusion strategy (see detailed discussions of different fusion strategies and definitions in Cohen et al. (1992) to combine data from video and LIDAR involves projecting trajectories from each sensor's respective sensor-centric coordinate system to a shared bird-eye-view (BEV) coordinate system of the intersection region (Figure 4-2). Vehicle trajectories are extracted from the fisheye video and pedestrian/bicyclist trajectories are extracted from the LIDAR, so that object trajectories are not assumed to overlap across sensors. We extract a 2D BEV map using Google maps of the intersection to use as a shared reference for both sensors. In Section 4.3.1 we described how a projection is estimated for mapping undistorted points from video frames to the BEV coordinate system. For the LIDAR sensor we first create a BEV representation of a target 3D point cloud by discarding the height (z) dimension. Given the reference image and the target BEV point cloud, landmarks are manually selected in both to compute a homography matrix which is used to rotate and translate point cloud coordinates onto the reference satellite image.

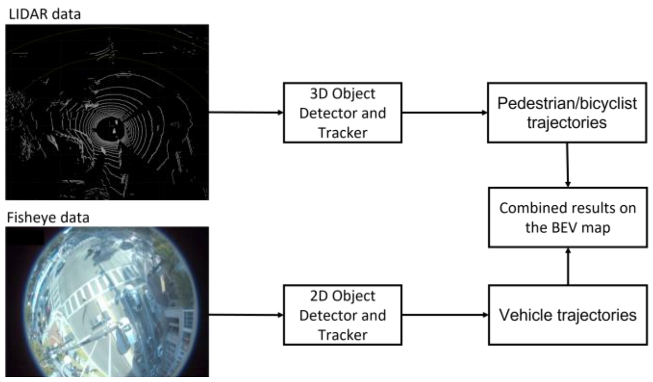
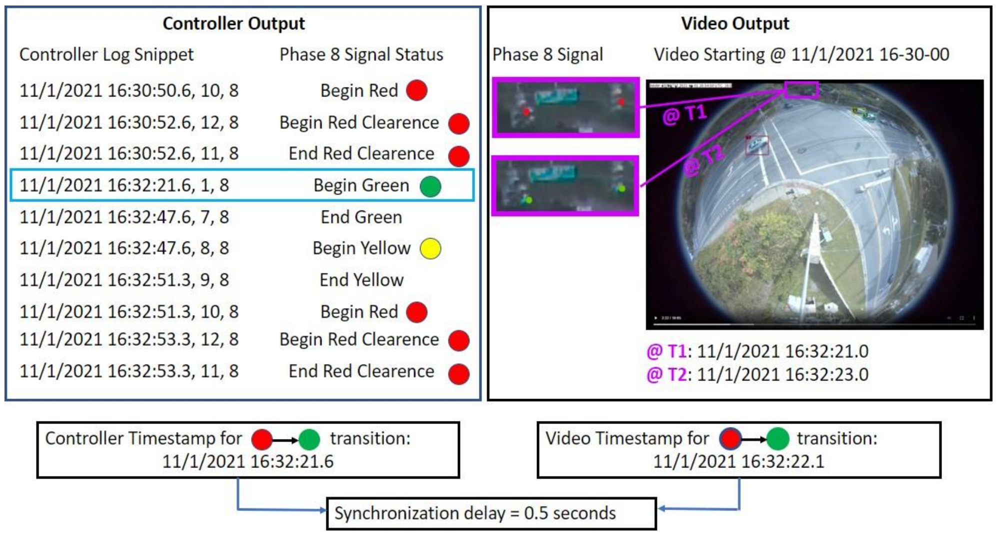

LIDAR data
Fisheye data
3D Object Detector and Tracker
2D Object Detector and Tracker
Pedestrian/bicyclist trajectories
Combined results on the BEV map
Vehicle trajectories

Figure 4-2. The pipeline of the fusion strategy of LIDARs and videos.

4.3.4. ATSPM Data Fusion

ATSPM data in the form of signal status for each signal head at an intersection is able to be accessed and stored in a database for analysis. For example, we use this information to assess red light violations for vehicles and walk signal violations for pedestrians. The challenge here is to align the timestamps of ATSPM data with that of the video and LIDAR data. This is handled by aligning signal change events of visible signal heads in fisheye video with the corresponding events in the ATSPM data (Figure 4-3).


Controller Output

Controller Log Snippet	Phase 8 Signal Status
11/1/2021 16:30:50.6, 10, 8	Begin Red 🔴
11/1/2021 16:30:52.6, 12, 8	Begin Red Clearence 🔴
11/1/2021 16:30:52.6, 11, 8	End Red Clearence 🔴
11/1/2021 16:32:21.6, 1, 8	Begin Green 🟢
11/1/2021 16:32:47.6, 7, 8	End Green
11/1/2021 16:32:47.6, 8, 8	Begin Yellow 🟡
11/1/2021 16:32:51.3, 9, 8	End Yellow
11/1/2021 16:32:51.3, 10, 8	Begin Red 🔴
11/1/2021 16:32:53.3, 12, 8	Begin Red Clearence 🔴
11/1/2021 16:32:53.3, 11, 8	End Red Clearence 🔴

Video Output

Phase 8 Signal   Video Starting @ 11/1/2021 16-30-00

@ T1: 11/1/2021 16:32:21.0
@ T2: 11/1/2021 16:32:23.0

Controller Timestamp for 🔴→🟢 transition: 11/1/2021 16:32:21.6

Video Timestamp for 🔴→🟢 transition: 11/1/2021 16:32:22.1

Synchronization delay = 0.5 seconds

Figure 4-3. Data collected from controller logs and video for computing synchronization delay between ATSPM and video data.

To automate this process, we use an image processing algorithm to extract the signal status from each visible signal head in the fisheye video. The algorithm expects the pixel region for each visible signal light to be provided as input. The average pixel value for red signal status and green signal status is estimated, and the current pixel values are compared against this average value to output a red or green status prediction for each video frame. The change in status is used to align the video frame timestamps with the ATSPM clock.

4.3.5. Clustering Trajectories by Phase

To provide a fine-grained understanding of road user behavior, we take the fused video and LIDAR trajectories for vehicles and pedestrians and cluster them by phase. To accomplish this, we use the hierarchical trajectory clustering methodology developed in our previous work (Banerjee et al., 2020). This method firstly clusters trajectories by movement direction and then uses a spectral clustering algorithm to cluster trajectories that all share the same direction of movement by phase. A robust and fast dynamic time warping distance measure is derived to compute the similarity between trajectories.

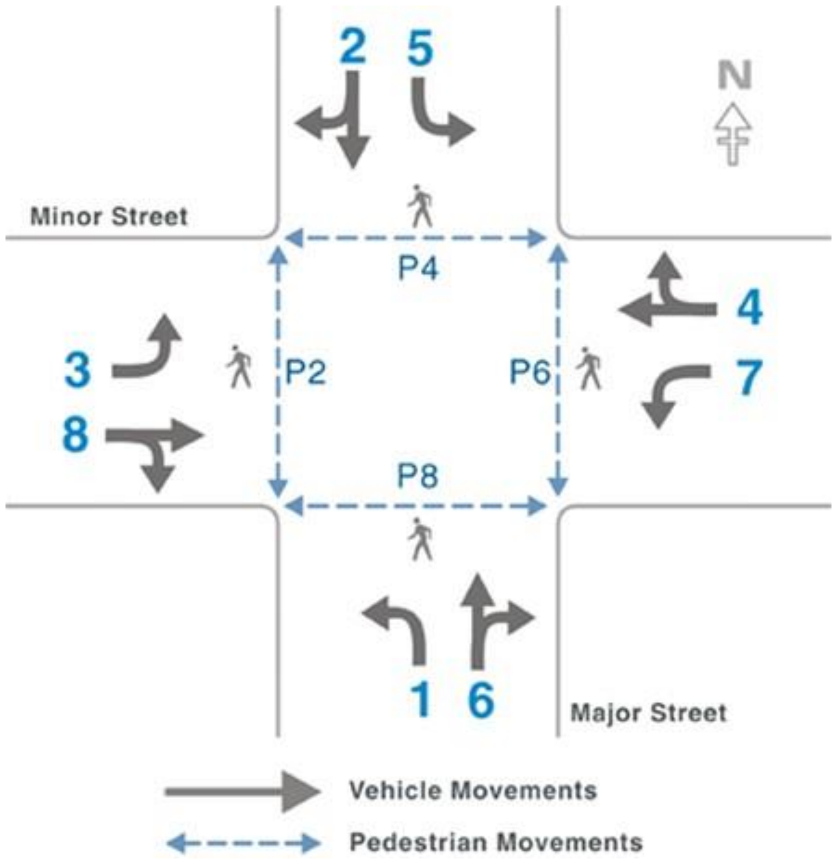


Figure 4-4. Signal phases for pedestrian and vehicle movements (Urbanik et al., 2015). The solid gray arrows show vehicle movements while the blue dotted arrows show pedestrian movements.

Phase Definitions and Terminology – Signal phases and their terminology are shown in Figure 4-4. The signal phases allow to accommodate the various users at intersections in a safe and efficient manner. Each phase is used as a specific timing unit to reflect certain one or more movements of road users such as vehicle and pedestrians as shown in Figure 4-4.

A signal phase and timing (SPaT) system records the state of each phase in terms of red, yellow, or green light at a certain frequency, which is achieved by the SpaT messages using a 24-bit binary number and its hexadecimal equivalent. Specifically, the first, second, and third of the eight bits are reserved for recording green, yellow, and red status, respectively. For example, if phases 2 and 6 are green, and every other signal is red, then, the 24-bit signal state is 0100 0100 0000 0000 1011 1011. The equivalent

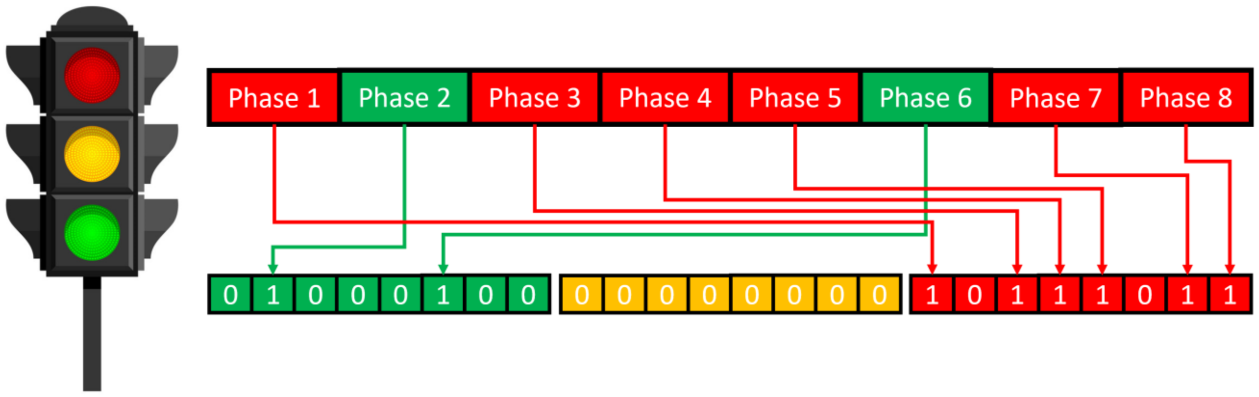


Figure 4-5. One visual example for the 24-bit signal state representation. For the eight phases, we color them to red, yellow, or green boxes to reflect their current status. As phases 2 and 6 are green, the first 8-bit for green has changed the 2nd and 6th bits to 1. As the rest phases are all in red, the third 8-bit for red has changed the corresponding bits to 1. The second 8-bit remain zeros as none of the eight phases are yellow.

hexadecimal representation is 4400bb. We provide the visual example in Figure 4-5. Once we obtain trajectories generated from both LIDAR and fisheye videos that is synchronized to the SPaT messages, we can attach each trajectory with a SpaT data and a certain phase, which helps to determine its current movement.

4.4. Multisensor Data Collection

We collected approximately one hour of LIDAR, fisheye video, and ATSPM data between 15:20 and 16:20, November 19th, 2021 at the intersection of NW 13th and University Ave in Gainesville, FL (Figure 4-6). This intersection is sufficiently busy to contain pedestrians and bicyclists in addition to heavy amounts of vehicle traffic. In addition, we collected another one hour of LIDAR, fisheye video, and ATSPM data between 19:30 and 20:30, December 17th, 2021 at the same intersection for nighttime. The LIDAR, video and ATSPM data is collected and processed as previously described. The sample frames of the collected fisheye videos at daytime and nighttime are shown in Figure 4-7.

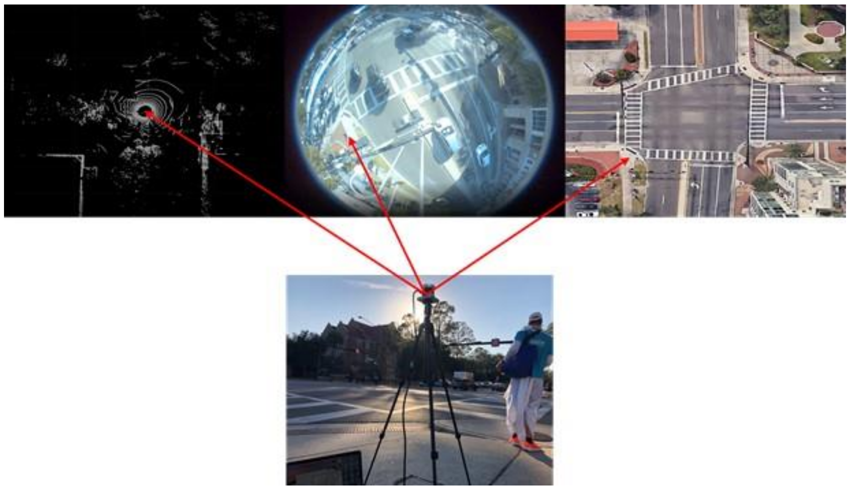


Figure 4-6. Sample frames of the collected fisheye videos at daytime (left) and nighttime (right).Figure 4 8. Multi-sensor data collection.

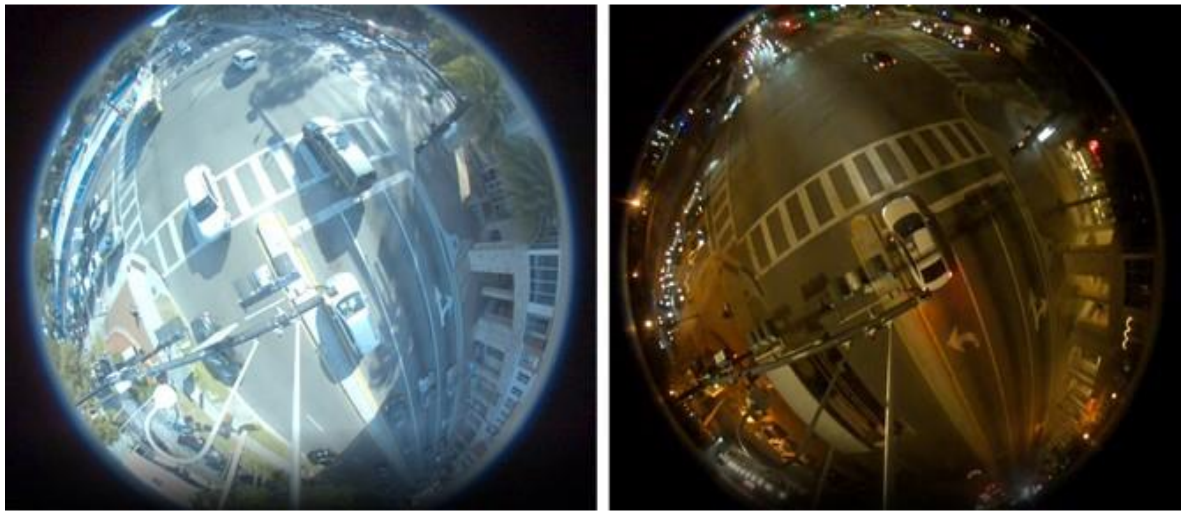


Figure 4-7. Sample frames of the collected fisheye videos at daytime (left) and nighttime (right).

4.5. Results

In this section we qualitatively analyze the results of the video, LIDAR, and ATSPM data fusion. We include results from sensor data collected at night, which allows us to highlight the benefits of using both video and LIDAR under limited visibility. To visualize the processed data, the trajectories from the LIDAR are drawn onto a 2D BEV reference image of the intersection alongside the fisheye camera trajectories. We organize trajectories by object class and by phase. The visualizations are obtained from the Web application developed for this LIDAR project. This involved updating the back-end database to accept trajectories originating from LIDAR as well as from fisheye cameras.

4.5.1. Pedestrian Detection at Nighttime

Pedestrian detection at nighttime using fisheye cameras is very challenging due to imbalanced illumination brought by the unstable light sources in nighttime surveillance, e.g., weak street lights and strong vehicle lights. Under this setting, the detector is struggling on tracking and detecting pedestrians (as shown in Figure 4-8). Fortunately, pedestrian detection is less impacted when moving to the LIDAR setting as LIDAR sensors remain effective at night (Figure 4-9). The project suggests that LIDAR sensors have the advantage on nighttime pedestrian detection over fisheye cameras.

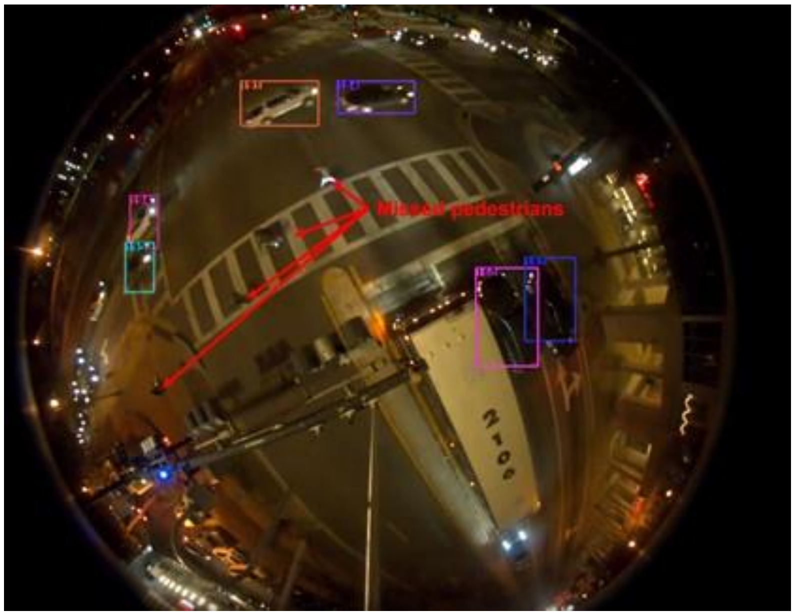


Figure 4-8. Examples of nighttime fisheye frames where the detector fails to detect pedestrians on the crosswalk.

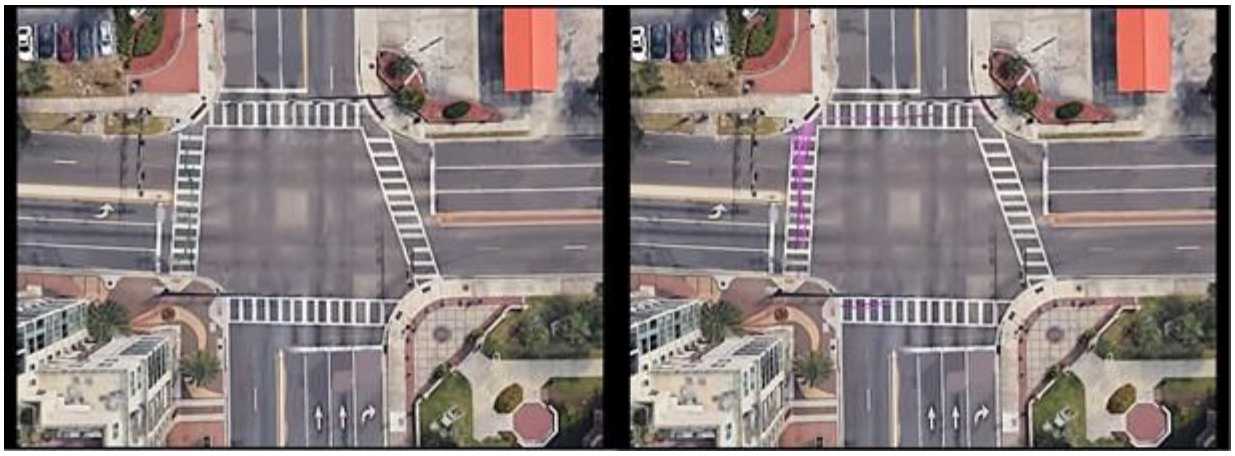
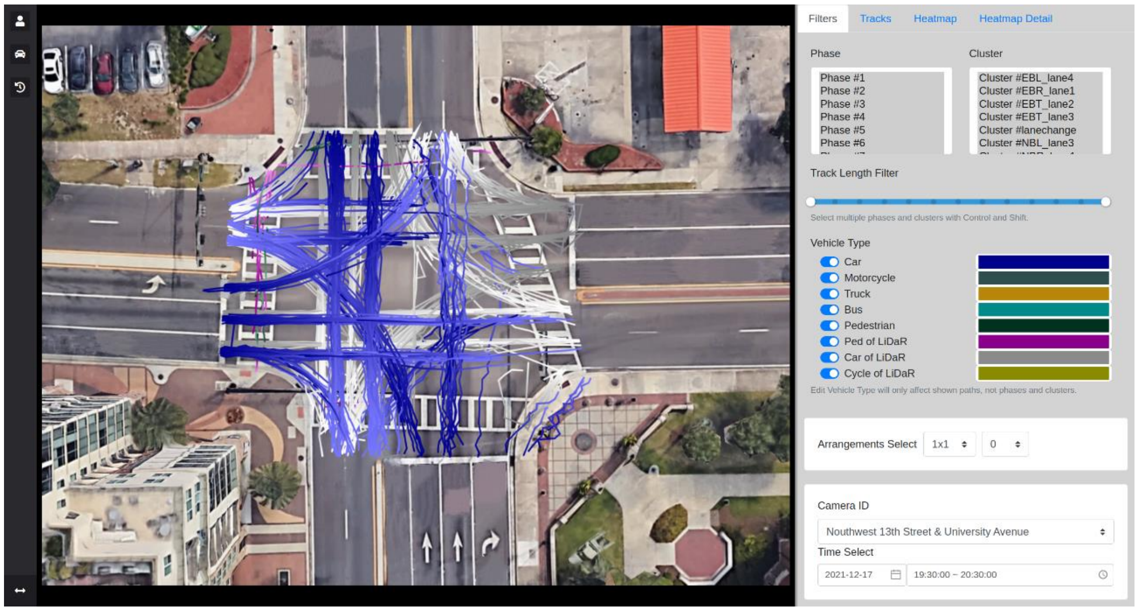


Figure 4-9. Comparisons between nighttime pedestrian detection using fisheye cameras (left) and LIDAR sensors (right) at the same time period. It is observed that pedestrian trajectories of the fisheye cameras are generally discontinuous, i.e., the whole trajectory of a single pedestrian might be broken down to several small trajectory segments due to the difficulty of consistently tracking and detecting a pedestrian at night. Trajectories from LIDAR videos are more complete. And it is able to detect pedestrians that are missed/dropped in the fisheye cameras.

4.5.2. Visualization of Generated Trajectories

The project provides a visualization of the generated trajectories based on time of occurrence, phase, and cluster. As shown in Figure 4-10, we choose a region of interest that focuses on the intersection region enclosed by the four crosswalks. Each trajectory is associated with a certain road user type, e.g., car, motorcycle, truck, bus, pedestrian, bicyclist, and is further assigned to a certain cluster distinguished by phase and behavior. Additional results are shown in Figures 4-12 and 4-13.



Car	🟦
Motorcycle	🟩
Truck	🟧
Bus	🟦
Pedestrian	🟩
Ped of LiDaR	🟪
Car of LiDaR	⬜
Cycle of LiDaR	🟨

Arrangements Select 1x1 0

Camera ID
Nouthwest 13th Street & University Avenue
Time Select
2021-12-17 19:30:00 ~ 20:30:00

Figure 4-10. Visualization of all detected trajectories from LIDAR and fisheye videos between 19:30 and 20:30, December 17th, 2021 at the same intersection. On the middle of the right panel, we provide check buttons for different types of trajectories with different types of trajectories with different colors.

4.5.3. Visualization of Trajectories Divided by Phase

Because the project has synchronized the ATSPM, video and LIDAR data source, it enables us to use the fused data for studying the behavior of road users with respect to the timing of an intersection. All trajectories in each time window may be further partitioned based on their movement, and/or cluster. As shown in Figure 4-11, all trajectories are partitioned into the eight phases discussed earlier in Figure 4-4, which is useful in sampling the major movements at an intersection. For example, as the pedestrian movement is synchronized to the ATSPM, the system can determine whether a walking pedestrian has violated a ped walk signal by checking its current phase number. The system can tell whether a vehicle is issuing a red-light violation by checking its current timestamp with the current timing of the intersection as well. Also, by counting trajectories in a certain time with the help of phase information, we can have an idea on the turning movement at the intersection.

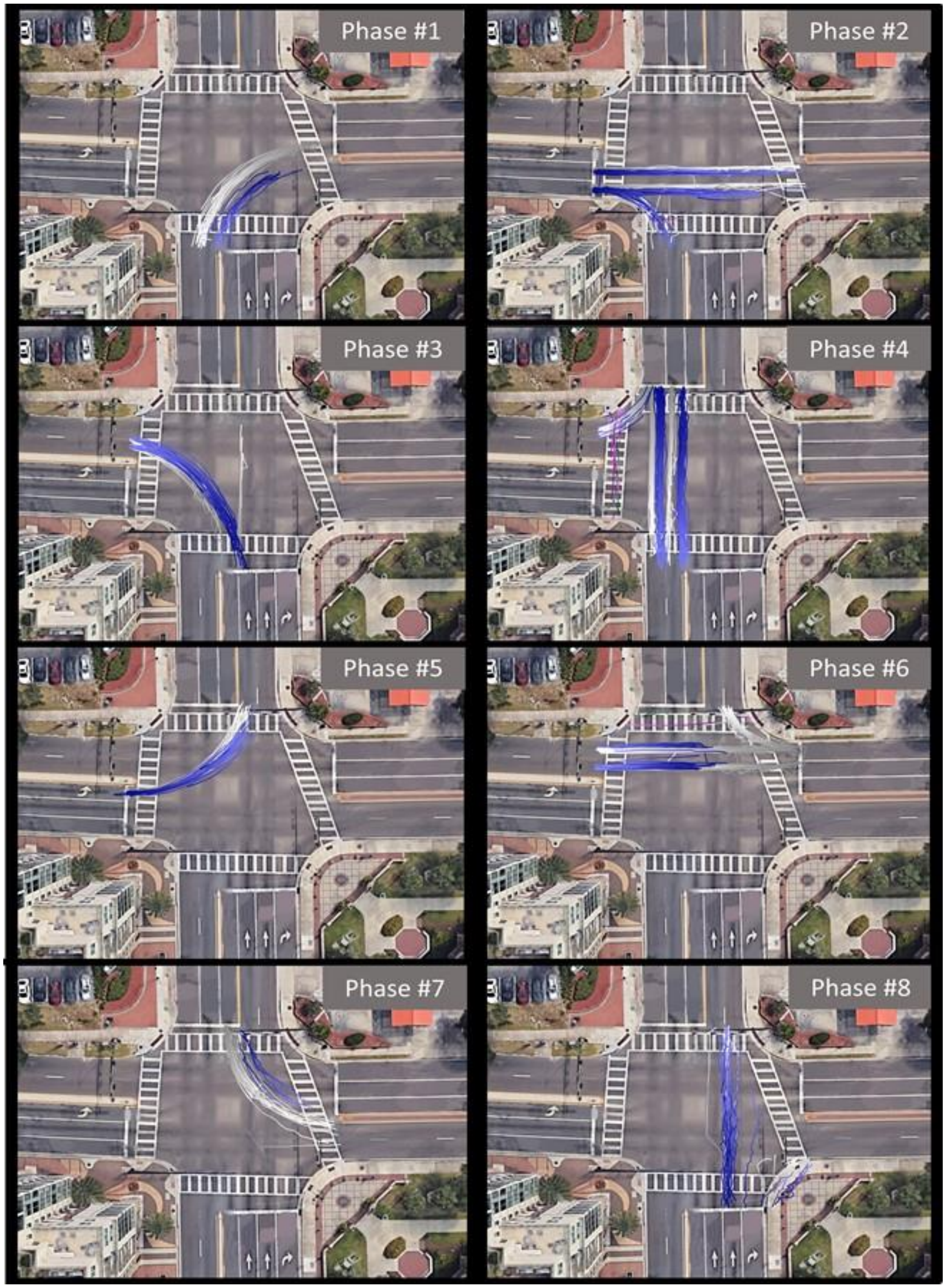
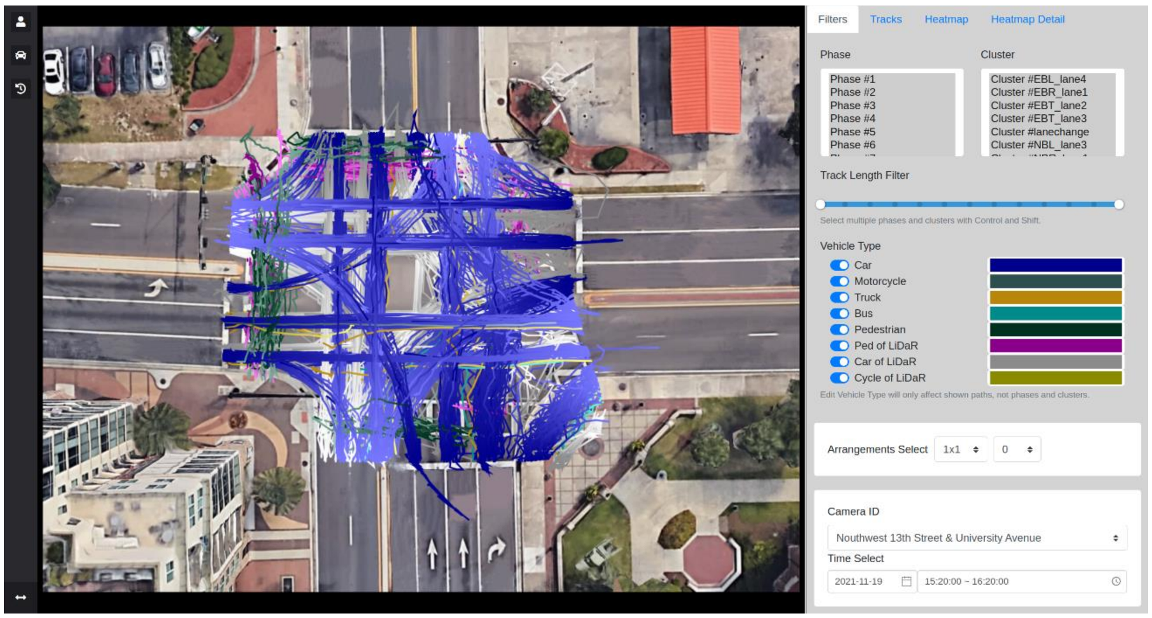


Figure 4-11. Visualization of all detected trajectories from LIDAR and fisheye videos at different phases. All tracks for vehicle movements on Friday, December 17, 2021, 7:30–8:30 PM, Eastern Time, are displayed.



Phase

Phase #1
Phase #2
Phase #3
Phase #4
Phase #5
Phase #6

Cluster

Cluster #EBL_lane4
Cluster #EBR_lane1
Cluster #EBT_lane2
Cluster #EBT_lane3
Cluster #lanechange
Cluster #NBL_lane3

Track Length Filter

Select multiple phases and clusters with Control and Shift.

Vehicle Type

Car	🟦
Motorcycle	🟩
Truck	🟧
Bus	🟦
Pedestrian	🟩
Ped of LiDaR	🟪
Car of LiDaR	⬜
Cycle of LiDaR	🟨

Edit Vehicle Type will only affect shown paths, not phases and clusters.

Arrangements Select 1x1 0

Camera ID

Northwest 13th Street & University Avenue

Time Select

2021-11-19 15:20:00 ~ 16:20:00

Figure 4-12. Visualization of all detected trajectories from LIDAR and fisheye videos between 15:20 and 16:20, November 19, 2021. On the middle of the right panel, we provide check buttons for different types of trajectories with different colors.

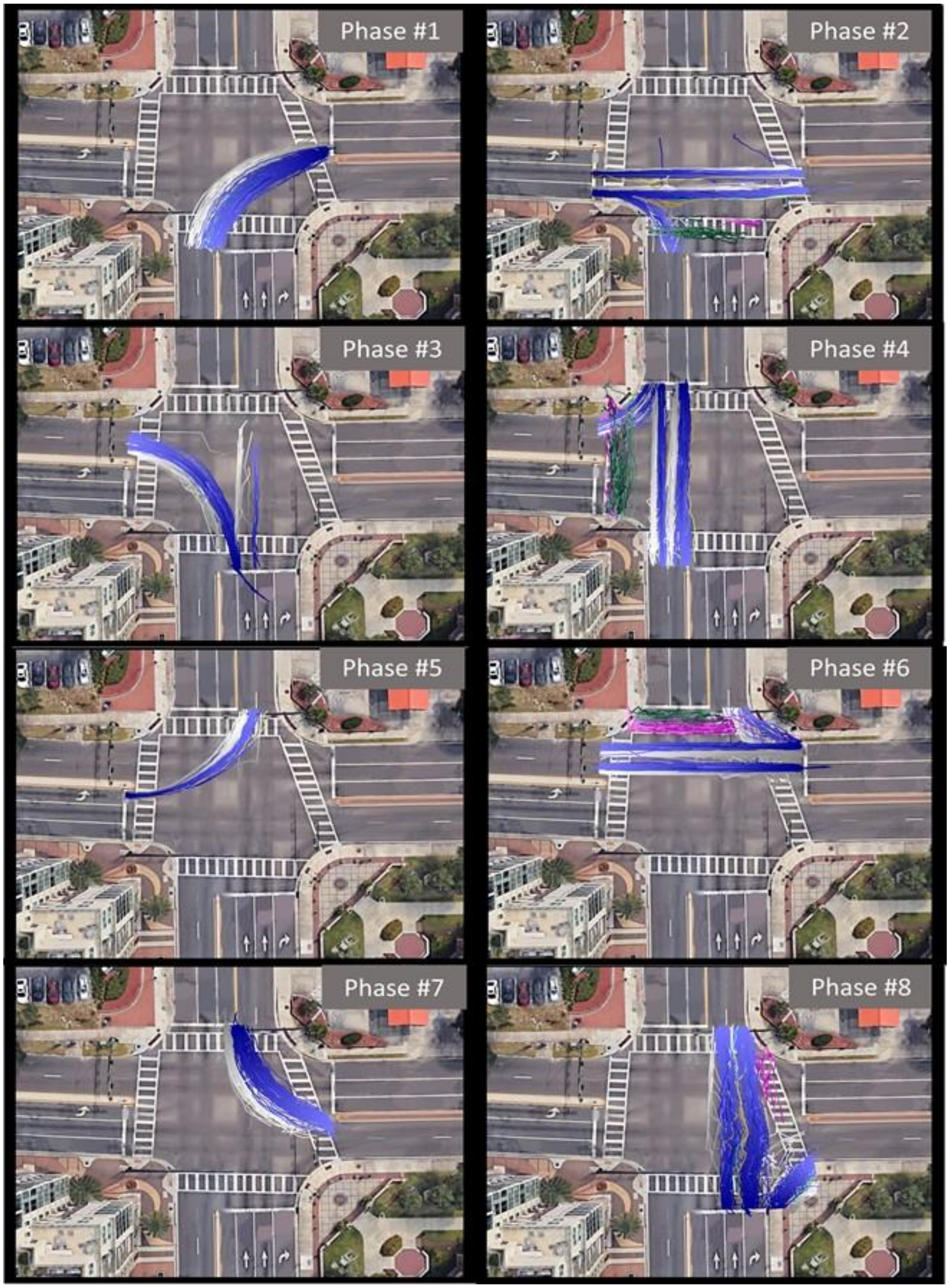


Figure 4-13. Visualization of all detected trajectories from LIDAR and fisheye videos at different phases. All tracks for vehicle movements on Friday, November 19, 2021, 3:20–4:20 PM, Eastern Time, are displayed.

4.6. Conclusions

The report has presented a system for detection, tracking, and clustering of traffic objects collected from fisheye cameras and LIDAR sensors. The algorithms for temporal synchronization of ATSPM, video, and LIDAR data sources have been developed to enable using the fused data to study the behaviors of road users with respect to the timing of intersections.

The approaches rely on recent advances in deep learning for 2D and 3D object detection, tracking and classification as well as conventional clustering methods. The project developed a fusion approach that combines trajectories with traffic signal timing data. From these, it can cluster trajectories based on its movement patterns and further linked them to traffic phases, which allows to associate each trajectory with a timing of an intersection reflecting certain movements.

The clustered trajectories from our developed algorithms provide insights on the moving patterns at the intersection – spatial locations and timestamps of all the objects passing through an intersection along with their object types are recorded and the object trajectories are clustered into typical movement patterns at intersections distinguished by factors such as traffic phase, abnormal behaviors. A traffic engineer can quickly analyze the normal clusters and anomalous clusters to get a better understanding of traffic behavior at an intersection.

In addition, we show that LIDAR sensors are more promising compared to cameras when moving to traffic object detection and tracking at nighttime and suggest that a wider deployment of LIDAR at intersection is beneficial to traffic surveillance. A LIDAR based solution can serve as a complement to the camera-based solutions to handle nighttime scenarios. The project also suggests a mounted location that is sufficiently high is necessary to reduce the occlusion created by objects approaching to the LIDAR sensor, which could further boost the performance of LIDAR detection and tracking. As LIDAR sensors become more accurate and cheaper, it is promising to be used as a reliable option at the edge in service of intelligent traffic intersection systems.

Chapter 5 – Develop a Trajectory Storage, Retrieval, and Visualization System

5.1. Introduction

Most traffic intersections today are equipped with sensors, which may include video cameras, LIDAR units, and/or traditional induction loop detectors. The advances in technology enable us to monitor these intersections remotely in real-time and identify potential inefficiencies. However, the high volume of data being collected by the sensors at each intersection and the number of such intersections in a city makes it imperative to automate the process of monitoring and analyzing problem areas. We have developed a visual analytics system to analyze the videos generated at an intersection using state-of-the-art machine learning algorithms and data communication mechanisms at the back end.

The system can detect movement conflicts or near-misses in streaming mode within a small latency. The system also has the capabilities to perform historical analysis where the user specifies a time window of interest, and the tool may filter and analyze the trajectories of vehicles and pedestrians passing through a traffic intersection during that time. The system is based on intersection videos obtained using fisheye cameras since fisheye cameras can capture a complete view of the intersection, unlike regular cameras that can record approaches from one direction.

The main contributions of our work presented in this report are summarized below.

- We developed a visualization system for the display and analysis of movement patterns of pedestrians and vehicles. This gives the user flexibility to filter trajectories based on the time and signaling phases of an intersection. The filtered trajectories, in combination with the ongoing signal timing information (STI) data, helps the user to determine the trajectories that violated the red light.
- It uses powerful clustering algorithms that segment the trajectories based on object type as well as path traversed, results in interesting visualization-based insights of traffic behavior in terms normal and safe versus rare and accident-prone. Besides, clustering also helps us determine counts of vehicles on different lanes as the trajectories on separate lanes form separate clusters.
- The system can be used to perform traffic analysis to get insights into key trends at an intersection. Being able to visualize the dominant traffic behavior in different settings may help the traffic engineer to address any potential bottlenecks by changing the timing plan or by revamping the intersection.

The rest of the report is organized as follows. Section 5.2 presents preliminary background on video processing. Section 5.3 presents an overview of the processes we use to unwarp the spherical images obtained from a fisheye camera, while Section 5.4 presents the core of our visualization software. Section 5.5 is devoted to the historical analysis mode of our visualization software, and Section 5.6 presents a trend analysis case study. We finally conclude in Section 5.7.

5.2. Background

In this section, we present some preliminary background of video processing, a fundamental component for trajectory generation. Figure 5-1 demonstrates the overall pipeline. First, we take the raw fisheye video as input and perform video processing using computer vision techniques for detecting and tracking road objects (such as cars and pedestrians). Then, we process and cluster trajectories for final visualization purposes. The various components used in this pipeline are described as follows.

Video Analysis – The raw video that is an input to our software is captured using fisheye cameras installed at traffic intersections. Compared to an ordinary video camera, a fisheye camera can capture the whole intersection in a wide panoramic and non-rectilinear image using its wide-angled fisheye lens. Fisheye cameras are advantageous because a single camera can capture a complete view of the entire intersection. If the intersection is large, two fisheye cameras may be installed to capture the complete intersection.

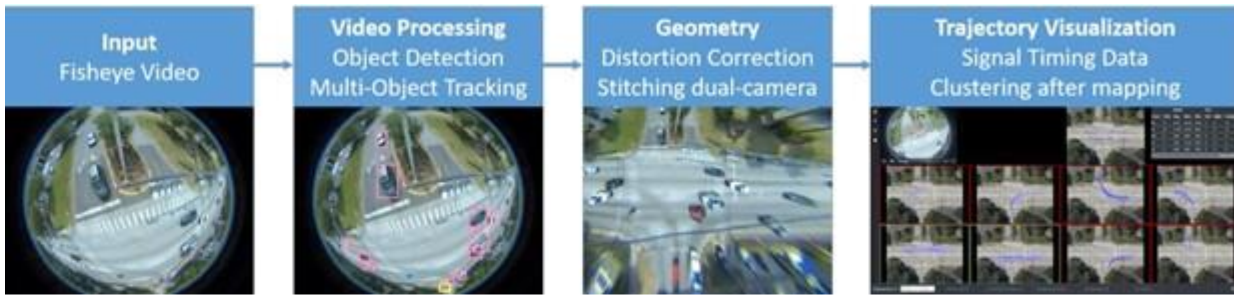
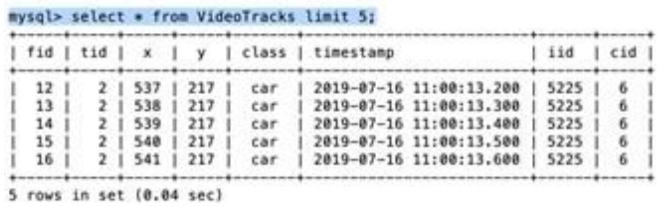


Figure 5-1. An overview of the pipeline consisting of video processing and multi-object tracking for trajectory generation followed by post processing and fusion with signal timing information and finally visualization

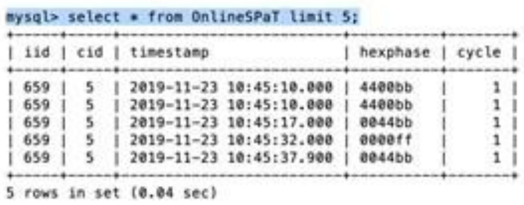
Processing the video obtained from a fisheye camera enables us to create trajectories of moving objects at an intersection. A trajectory of a moving object is its path represented by timestamped location coordinates of the object. For a typical, moderately busy intersection we studied, the volume of traffic is enormous, with over 10,000 trajectories being generated on a weekday. Thus, a visualization system is critical to understand the traffic behavior for a given intersection for a specific period. For privacy protection, the information about the moving object is automatically anonymized by saving only the location coordinates of objects and, in the cases of vehicles, their size and color to our database.

Video processing generates frame-by-frame detection and tracking of all the moving objects in an intersection. It also uses a temporal superpixel (supervoxel) method (Huang et al., 2018) to extract an accurate mask for object representation. These can be converted into trajectories that represent the spatial and temporal movement of traffic. A trajectory is a path traversed by a moving object that is represented as successive spatial coordinates and corresponding timestamps. Details of the video processing and analysis are described briefly below and provided in detail in (Huang et al., 2020).

Description of Database – The trajectories generated by video processing are uploaded in a MySQL database on the cloud. STI information is extracted from high-resolution controller logs and stored is a separate database as well. Figure 5-2 presents the key attributes. The fields of a single row in TrackInfo are: frame ID, which identifies the current video frame; track ID, which identifies a trajectory; x and y are the coordinates of the object location; w and h are the width and height of the bounding box enclosing the object; intersection ID identifies a specific intersection; timestamp, phase-encoded in hexadecimal format, and the cycle number, with respect to the first observation. The camera ID is an extra field for later use. It is possible to derive additional information such as class or type of object (car, bus, truck, motorbike, pedestrian), the intersection, date, and time represent the timestamp.



```
mysql> select * from VideoTracks limit 5;
+-----+-----+-----+-----+-------+-------------------------+------+-----+
| fid | tid |  x  |  y  | class | timestamp               | iid  | cid |
+-----+-----+-----+-----+-------+-------------------------+------+-----+
|  12 |   2 | 537 | 217 |  car  | 2019-07-16 11:00:13.200 | 5225 |  6  |
|  13 |   2 | 538 | 217 |  car  | 2019-07-16 11:00:13.300 | 5225 |  6  |
|  14 |   2 | 539 | 217 |  car  | 2019-07-16 11:00:13.400 | 5225 |  6  |
|  15 |   2 | 540 | 217 |  car  | 2019-07-16 11:00:13.500 | 5225 |  6  |
|  16 |   2 | 541 | 217 |  car  | 2019-07-16 11:00:13.600 | 5225 |  6  |
+-----+-----+-----+-----+-------+-------------------------+------+-----+
5 rows in set (0.04 sec)
```



```
mysql> select * from OnlineSPaT limit 5;
+-----+-----+-------------------------+----------+-------+
| iid | cid | timestamp               | hexphase | cycle |
+-----+-----+-------------------------+----------+-------+
| 659 |  5  | 2019-11-23 10:45:10.000 | 4400bb   |     1 |
| 659 |  5  | 2019-11-23 10:45:10.000 | 4400bb   |     1 |
| 659 |  5  | 2019-11-23 10:45:17.000 | 0044bb   |     1 |
| 659 |  5  | 2019-11-23 10:45:32.000 | 0000ff   |     1 |
| 659 |  5  | 2019-11-23 10:45:37.900 | 0044bb   |     1 |
+-----+-----+-------------------------+----------+-------+
5 rows in set (0.04 sec)
```

Figure 5-2. MySQL databases of trajectories generated by video processing (left) and the signal timing information (right) generated from ATSPM.

Trajectory Processing – In this section, we present the steps and methods for trajectory generation and processing. First, for lower computation cost, we process each trajectory to eliminate redundant coordinates. Then, we integrate the signal timing data into track data, where signal timing data can provide additional cues for trajectory analysis. Finally, we use an unsupervised approach to cluster the trajectories that can be used for mining.

We reduce the number of coordinates in each trajectory with minimal loss of information, which is done in a top-down manner using the Douglas-Peucker algorithm, also known as the iterative end-point fit algorithm (Douglas and Peucker, 1973; Ramer, 1972). For a given trajectory T, the Douglas-Peucker algorithm finds another trajectory, T_0, with fewer coordinates, such that the maximum distance between T and T_0 is below a certain threshold, e, as specified by a user.

Fusion with Signal Timing – Trajectories at intersections are dictated by the ongoing signaling status at an intersection. With the availability of advanced controllers that can record signal changes and detector events at a very high resolution (10 Hz), it is possible to generate a signal phase and timing log for the intersection for a given period.

A signal timing information (STI) system record the state of each phase in terms of red, yellow, or green light at a certain frequency. In our application, we record STI messages from high-resolution data on a decisecond basis (10 Hz), using a 24-bit binary number and its hexadecimal equivalent. Out of the 24 bits, the first, second, and third of the eight bits are reserved for recording green, yellow, and red status, respectively. For example, if phases 2 and 6 are green, and every other signal is red, then, the 24-bit signal state is 0100 0100 0000 0000 1011 1011. The equivalent hexadecimal representation is 4400bb.

The availability of video data along with high-resolution controller data enables us to place the trajectories from video data in the context of the signal phases. As we get a new video, one of the first tasks is to synchronize the beginning of the video with the timestamp on the controller logs, which can be done automatically with the help of this tool, the details of which are presented later in the paper. After synchronizing the beginning of the video with the corresponding timestamp on the STI data, we join the STI data with the trajectory data based on the timestamp for each of the remaining frames.

Clustering Trajectories – We compute the distance between two trajectories using FastDTW (Salvador and Chan, 2004) and a method that computes the area between corresponding points in two trajectories. FastDTW approximates the Dynamic Time Warping algorithm used to find the optimal alignment between two-time series with near-linear time and space complexity. It is appropriate to apply it to find the spatial distance between trajectories because objects are traversing their respective trajectories at different speeds.

We have developed a variation of the K-means algorithm for clustering and over-cluster to ensure that the clusters generated have high similarity. A post-processing step is often needed to merge these clusters as multiple clusters created for the same movement. Multiple clusters are created because trajectories sometimes can be truncated early or may start late due to occlusion on the scene.

Our visualization software is deployed on the cloud. The core processing steps, namely, video analysis, trajectory processing, fusion with signal timing and clustering trajectories were designed to happen at an edge or intermediate server to minimize latency issues, and the data generated by the processing steps are stored on the cloud.

5.3. Visualization

Our visualization software is divided into two parts, the frontend, and the backend. The front end is written in Vue.js, which is one of the most popular progressive frameworks for building user interfaces. The frontend has two modes, the streaming mode, and the historical mode. The streaming mode loads the video and corresponding starting timestamp from the backend, then uses a cache to query the database every 5 seconds for three operations, namely, track animation, display of the statistics related to the tracks, and the display of tracks by phases. In the historical mode, on the other hand, the software reads the database based on filters that the user can select. The backend is written in Node.js. The primary purpose of the backend is to query the database. The backend loads the Ajax post from the frontend, then queries the database using the received data. The backend also prepares the video file or video stream to be displayed in the frontend. We now describe the capabilities of our software in more detail, first for the streaming mode visualization in this section and then describe the capabilities of our software in more detail, and then for historical analysis in Section 5.5. The overall structure of the software can be described using a Data Flow Diagram (DFD). The diagram in Figure 5-3 shows how the data goes through the pipeline and finally appears in the interface.

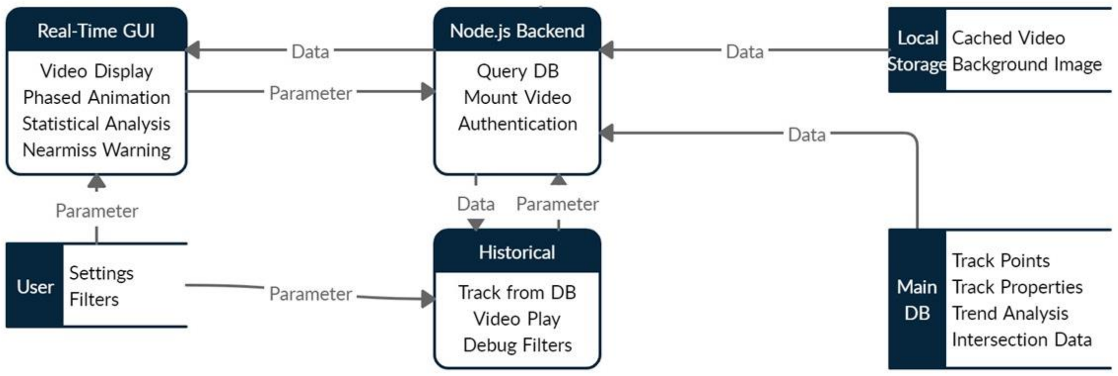
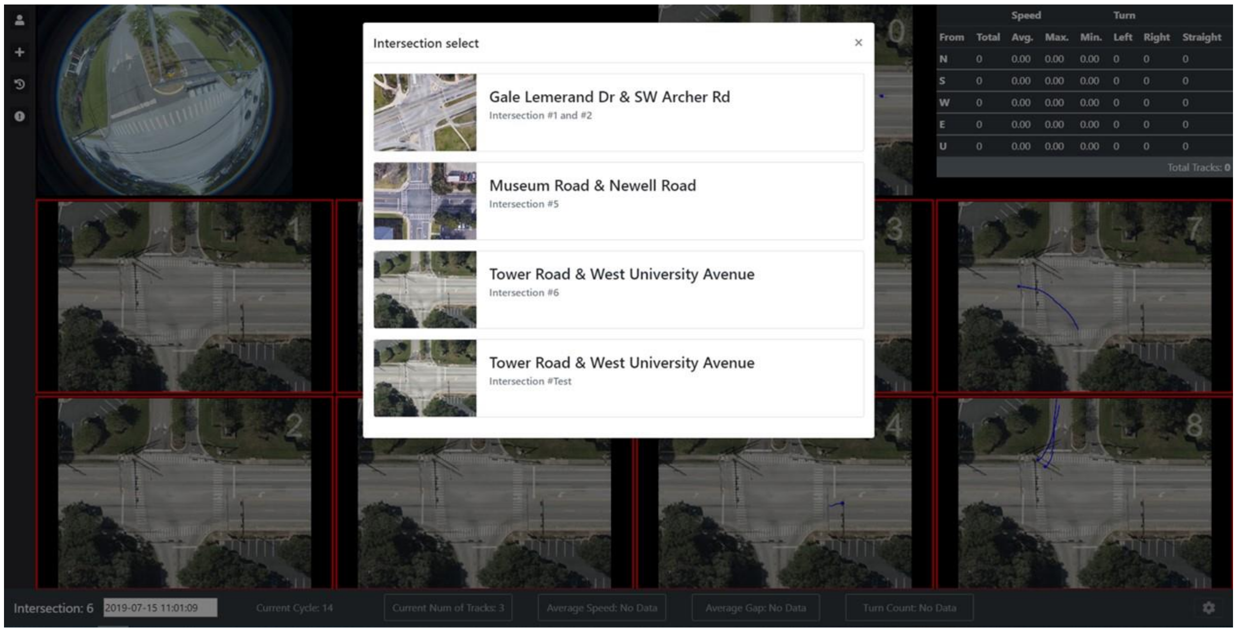


Figure 5-3. Data flow diagram of the whole process

Video Display – The fisheye video of the intersection, as captured by a fisheye camera is streamed or played on the top left corner of the webpage. An animation of the object trajectories is created by representing the objects as moving dots at the intersection. This animation is displayed beside the actual video in an unwarped space and in a time-synchronized manner for the actual video. This is shown in Figures 5-4 and 5-5.



Intersection select ×

Gale Lemerand Dr & SW Archer Rd
Intersection #1 and #2

Museum Road & Newell Road
Intersection #5

Tower Road & West University Avenue
Intersection #6

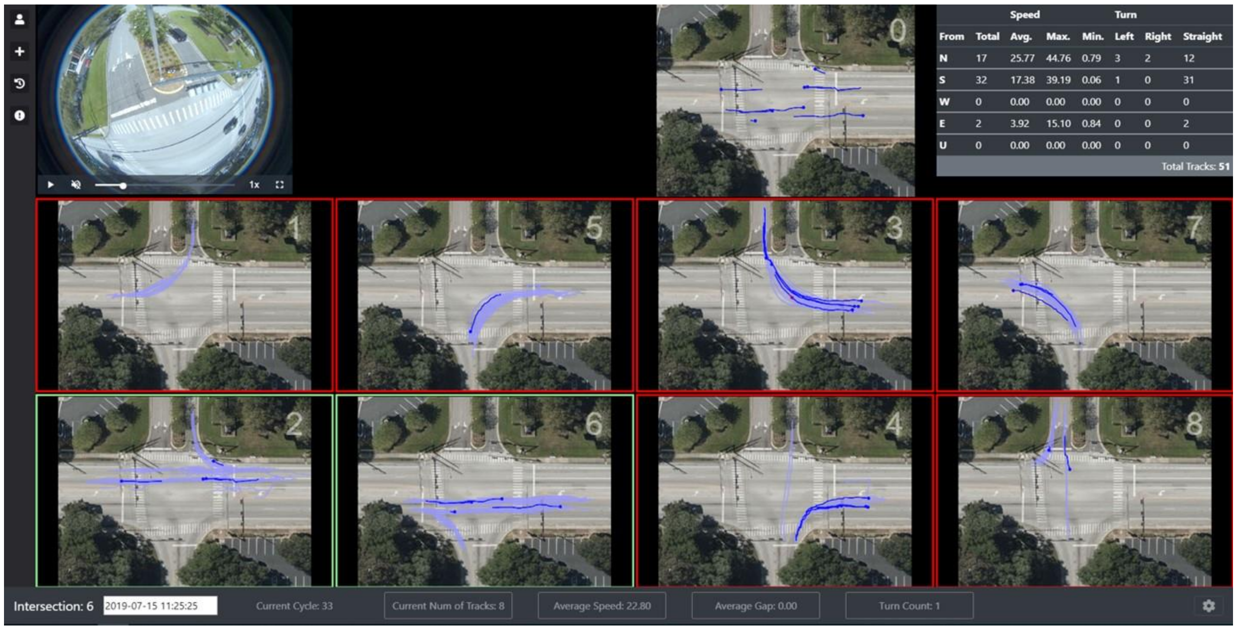
Tower Road & West University Avenue
Intersection #Test

		Speed			Turn		
From	Total	Avg.	Max.	Min.	Left	Right	Straight
N	0	0.00	0.00	0.00	0	0	0
S	0	0.00	0.00	0.00	0	0	0
W	0	0.00	0.00	0.00	0	0	0
E	0	0.00	0.00	0.00	0	0	0
U	0	0.00	0.00	0.00	0	0	0

Total Tracks: 0

Intersection: 6 2019-07-15 11:01:09 Current Cycle: 14 Current Num of Tracks: 3 Average Speed: No Data Average Gap: No Data Turn Count: No Data

Figure 5-4. *The selection of current intersection can be opened by clicking on the button on the sidebar.*



		Speed			Turn		
From	Total	Avg.	Max.	Min.	Left	Right	Straight
N	17	25.77	44.76	0.79	3	2	12
S	32	17.38	39.19	0.06	1	0	31
W	0	0.00	0.00	0.00	0	0	0
E	2	3.92	15.10	0.84	0	0	2
U	0	0.00	0.00	0.00	0	0	0

Total Tracks: 51

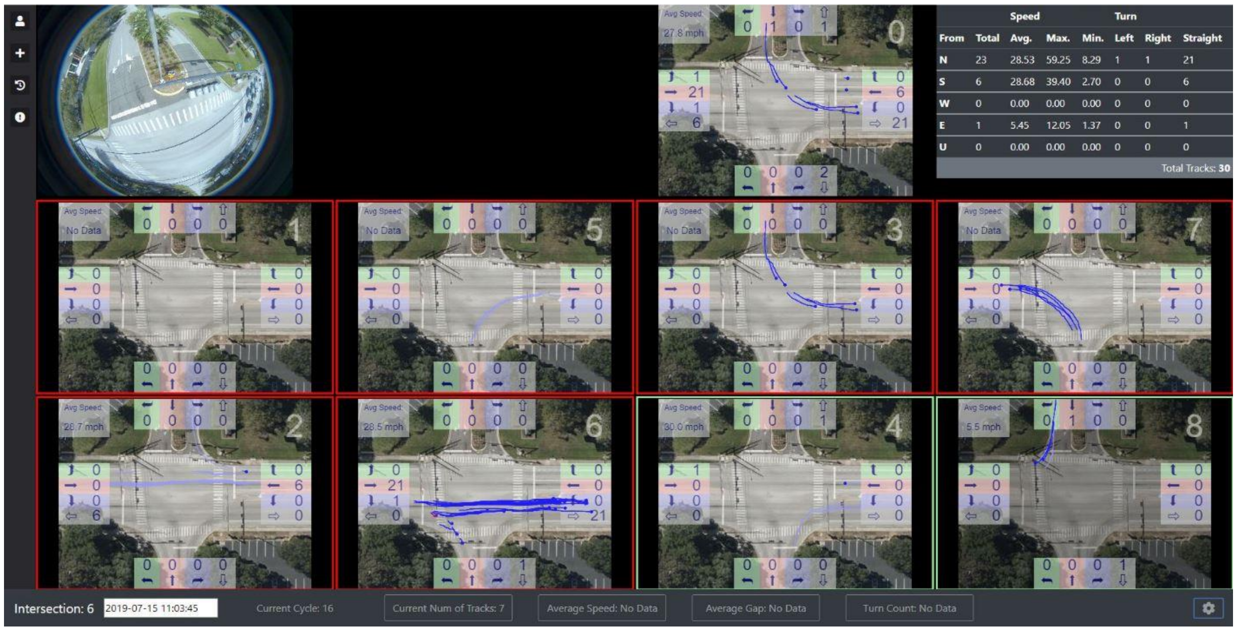
Intersection: 6 2019-07-15 11:25:25 Current Cycle: 33 Current Num of Tracks: 8 Average Speed: 22.80 Average Gap: 0.00 Turn Count: 1

Figure 5-5. Interface for showing the tracks. The video from the fisheye camera is at the top left corner. This is followed by an unwarped image of the intersection and the trajectories as represented by a moving dot. The statistics of the trajectories on the intersection are captured in the table on the top right side. The trajectories are partitioned into phases and displayed in the eight windows. The setting options and average statistics about the trajectories are present at the bottom.

Each point in the fisheye image is mapped onto the unwarped image using the mapping functions described in Section 5.3. For streaming mode accessibility of the data, we use database caches or put the data in memory where appropriate.

Phase Windows – There is a window for each of the eight phases of traffic movement, as shown in Figure 5-6. The eight windows are tiled and numbered according to the phase they represent. An ongoing trajectory is assigned a phase by first performing matches with the existing centroids or cluster centers. The phase of the trajectory is set to the phase of the matching centroid. After the phase is assigned, the trajectory is displayed in the respective phase window. Separating the trajectories by phases not only makes the trajectories easy to track but also helps us count the number of vehicles and their classes for each movement. The software provides support for an overlay on the phase windows that displays the number of vehicles in the through and turn lanes for that movement for a given time window, which is shown in the Figure 5-7. The user may enable the overlay by first clicking on the settings and then toggling the "Enable Overlay" switch.

Integration with STI – The integration of the trajectories with STI data is done in our software to place the trajectories in the context of the current signaling state of the intersection. This STI integration enables



		Speed			Turn		
From	Total	Avg.	Max.	Min.	Left	Right	Straight
N	23	28.53	59.25	8.29	1	1	21
S	6	28.68	39.40	2.70	0	0	6
W	0	0.00	0.00	0.00	0	0	0
E	1	5.45	12.05	1.37	0	0	1
U	0	0.00	0.00	0.00	0	0	0

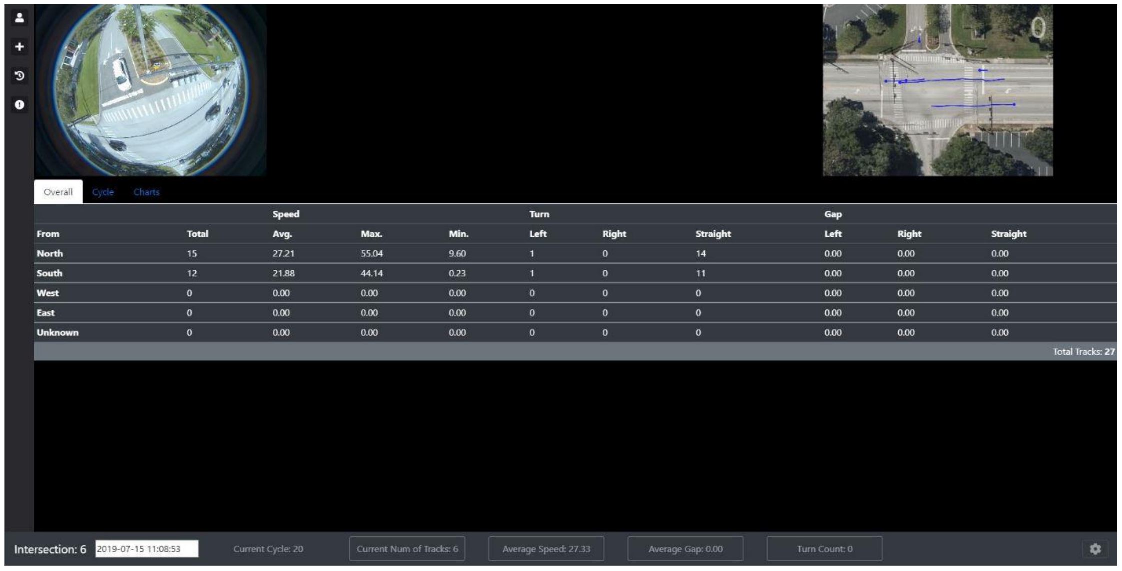
Total Tracks: **30**

Intersection: 6 2019-07-15 11:03:45 Current Cycle: 16 Current Num of Tracks: 7 Average Speed: No Data Average Gap: No Data Turn Count: No Data

Figure 5-6 The overlay feature of the tool may be used to display the vehicle egress and ingress counts for the different phases of movement.

us to detect anomalous traffic behavior at the intersection, such as vehicles moving in the yellow signals or those jumping the red. The display of the STI information in the GUI is done by framing each window with a box of the same color as that of the signal for that phase. The phases 2 and 6 are being served in Figure 5-7, and hence these phase windows are framed by green boxes in the figure.

Statistics – The statistics for the various trajectories are available in the form of charts and tables. On the streaming page of Figure 5-7, the table appears on the top right corner. The table shows the most common statistics, such as the count of the vehicles or pedestrians for each movement, and the average, minimum, and maximum speeds in mph for those movements. If the user is interested in more details such as the average gap or the response time, the table may be clicked to bring up the details. The further details are organized into three tabs. These are namely, Overall, Cycle and Charts, as shown in the Figures 5-8, 5-9, and 5-10, respectively. The Overall tab gives the overall observations, as the name suggests. The Cycle tab aggregates the data on a cycle by cycle basis. The Chart tab, on the other hand, shows the overall trend of the characteristics over time.

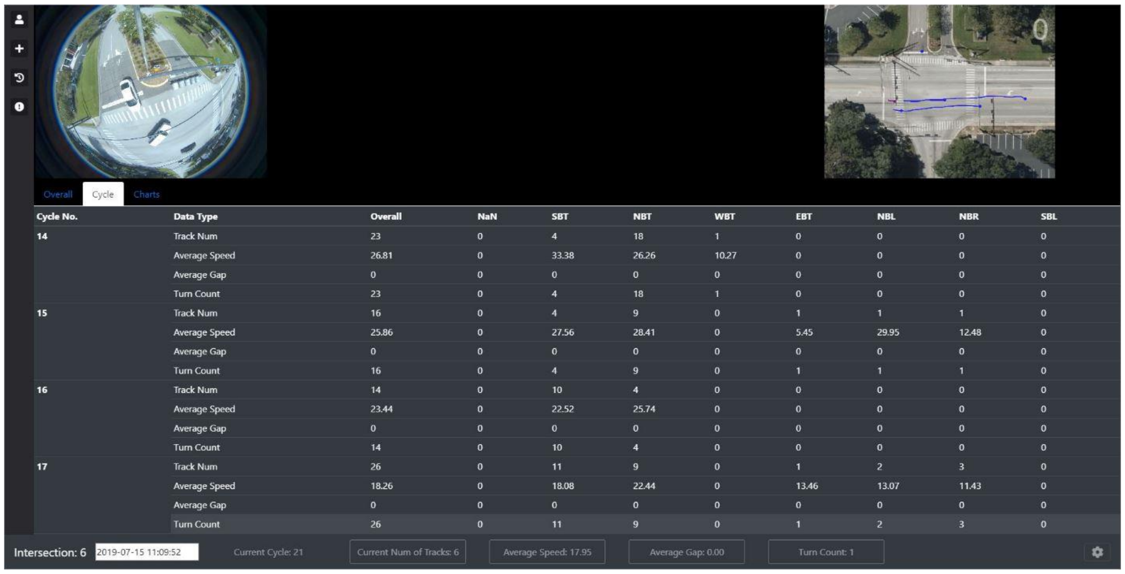


		Speed			Turn			Gap		
From	Total	Avg.	Max.	Min.	Left	Right	Straight	Left	Right	Straight
North	15	27.21	55.04	9.60	1	0	14	0.00	0.00	0.00
South	12	21.88	44.14	0.23	1	0	11	0.00	0.00	0.00
West	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00
East	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00
Unknown	0	0.00	0.00	0.00	0	0	0	0.00	0.00	0.00

Total Tracks: **27**

Intersection: 6 2019-07-15 11:08:53 Current Cycle: 20 Current Num of Tracks: 6 Average Speed: 27.33 Average Gap: 0.00 Turn Count: 0

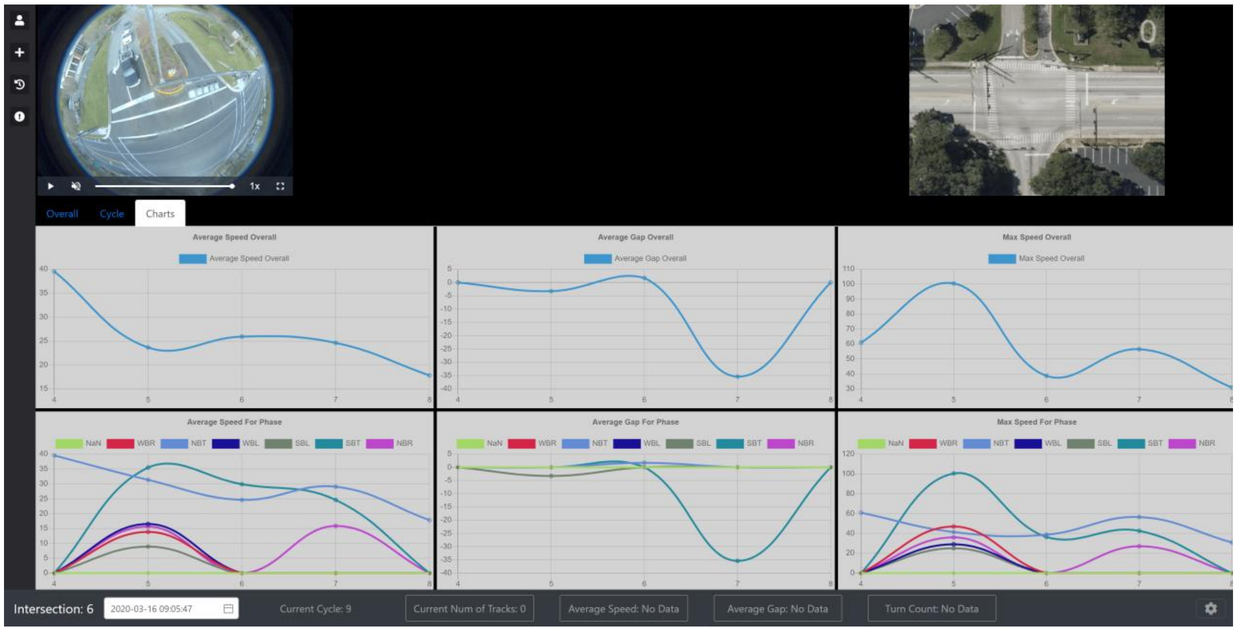
Figure 5-7. Clicking on the statistics table on the main page toggles the display to more statistical details. These are organized into three tabs, namely, Overall, Cycle, and Charts. We see the overall statistics here, for speed, gap and turn movement counts.



Cycle No.	Data Type	Overall	NaN	SBT	NBT	WBT	EBT	NBL	NBR	SBL
14	Track Num	23	0	4	18	1	0	0	0	0
	Average Speed	26.81	0	33.38	26.26	10.27	0	0	0	0
	Average Gap	0	0	0	0	0	0	0	0	0
	Turn Count	23	0	4	18	1	0	0	0	0
15	Track Num	16	0	4	9	0	1	1	1	0
	Average Speed	25.86	0	27.56	28.41	0	5.45	29.95	12.48	0
	Average Gap	0	0	0	0	0	0	0	0	0
	Turn Count	16	0	4	9	0	1	1	1	0
16	Track Num	14	0	10	4	0	0	0	0	0
	Average Speed	23.44	0	22.52	25.74	0	0	0	0	0
	Average Gap	0	0	0	0	0	0	0	0	0
	Turn Count	14	0	10	4	0	0	0	0	0
17	Track Num	26	0	11	9	0	1	2	3	0
	Average Speed	18.26	0	18.08	22.44	0	13.46	13.07	11.43	0
	Average Gap	0	0	0	0	0	0	0	0	0
	Turn Count	26	0	11	9	0	1	2	3	0

Intersection: 6 2019-07-15 11:09:52 Current Cycle: 21 Current Num of Tracks: 6 Average Speed: 17.95 Average Gap: 0.00 Turn Count: 1

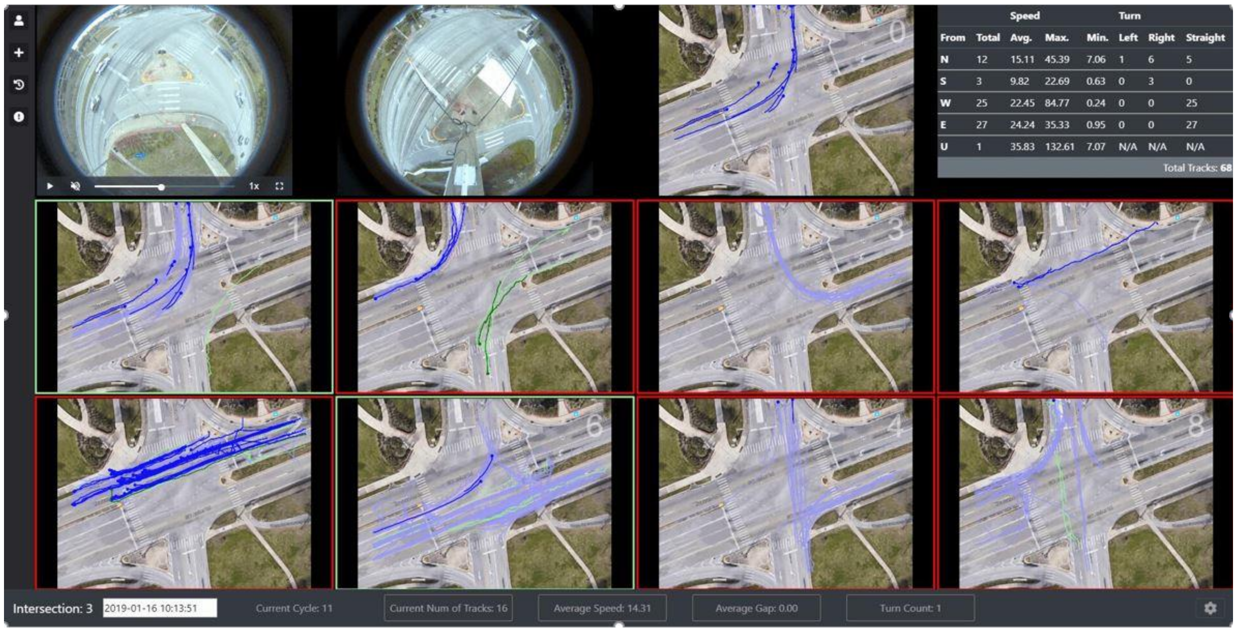
Figure 5-8 The cycle tab of the detailed statistical data shows the statistics data aggregated by signal cycles and displayed by cycle counts.



Intersection: 6 2020-03-16 09:05:47 Current Cycle: 9 Current Num of Tracks: 0 Average Speed: No Data Average Gap: No Data Turn Count: No Data

Figure 5-9. The charts tab when clicked shows the speed and gap statistics visually as charts.

Multicamera Support – Our visualization tool displays a merged image from two or more cameras that are typically installed in a large intersection. Figure 5-10 shows one such intersection, where two cameras are installed in the opposite corners of a large intersection. The animation merges the views from the two cameras and creates a single image.



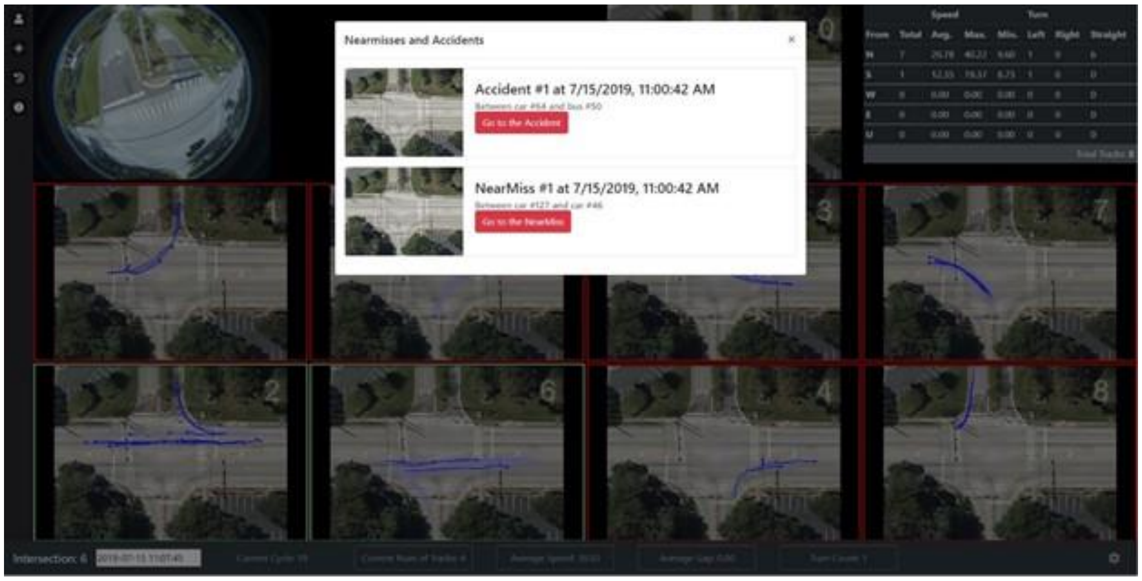
		Speed			Turn		
From	Total	Avg.	Max.	Min.	Left	Right	Straight
N	12	15.11	45.39	7.06	1	6	5
S	3	9.82	22.69	0.63	0	3	0
W	25	22.45	84.77	0.24	0	0	25
E	27	24.24	35.33	0.95	0	0	27
U	1	35.83	132.61	7.07	N/A	N/A	N/A

Total Tracks: 68

Intersection: 3 2019-01-16 10:13:51 Current Cycle: 11 Current Num of Tracks: 16 Average Speed: 14.31 Average Gap: 0.00 Turn Count: 1

Figure 5-10. The images from an intersection with two cameras. The cameras are installed in opposite corners of the intersection and the raw video from each of the cameras appears on the top left corner.

Near-miss and Crash Events – The near-miss and crash events are picked up by our tool automatically and reported separately. Such incidents are normally rare, but Figure 5-11 is a representation of a hypothetical situation to demonstrate the feature.



Accident #1 at 7/15/2019, 11:00:42 AM
Between car #64 and bus #50

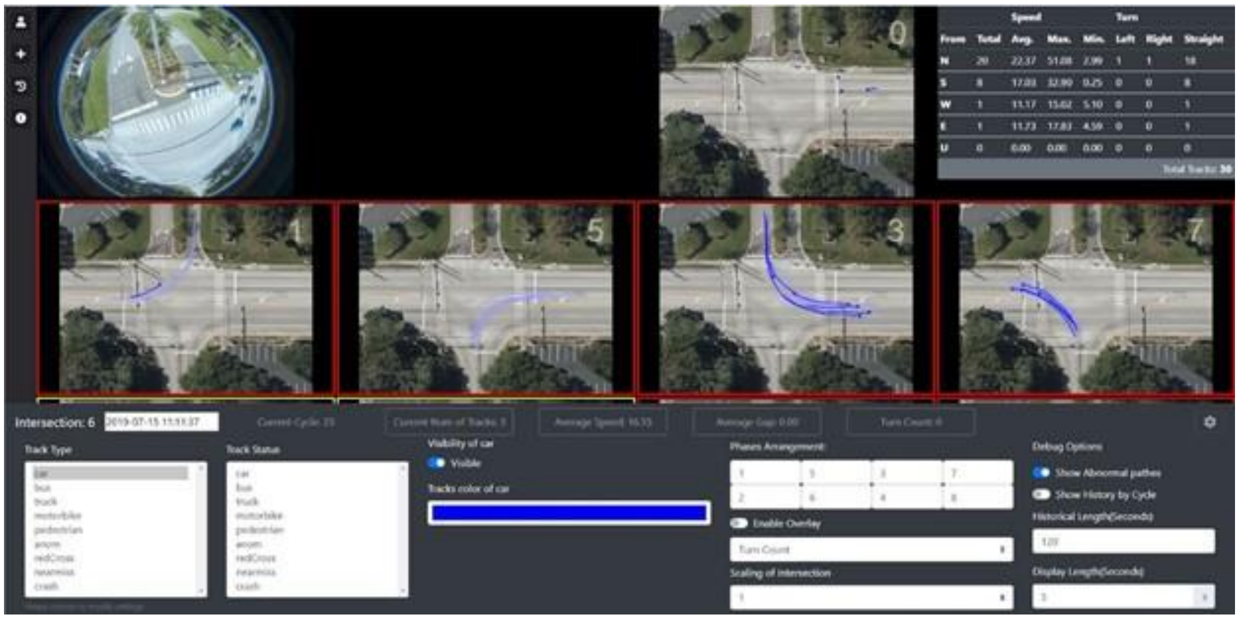
NearMiss #1 at 7/15/2019, 11:00:42 AM
Between car #127 and car #46

		Speed			Turn		
From	Total	Avg.	Max.	Min.	Left	Right	Straight
N	7	26.78	40.22	9.60	1	0	6
S	1	52.55	78.37	8.75	1	0	0
W	0	0.00	0.00	0.00	0	0	0
E	0	0.00	0.00	0.00	0	0	0
U	0	0.00	0.00	0.00	0	0	0

Total Tracks: 8

Figure 5-11. The interface for displaying near-miss and crash events in streaming mode. The two events described here are hypothetical and are used to test the interface.

Settings – The settings menu (Figure 5-12) provides access to the features described and other useful features, such as coloring trajectories by object class or selecting the time frame for displaying tracks, e.g., tracks that happened in the last x seconds. Figure 5-13 shows an example of this menu being used to get the details of an ongoing trajectory.



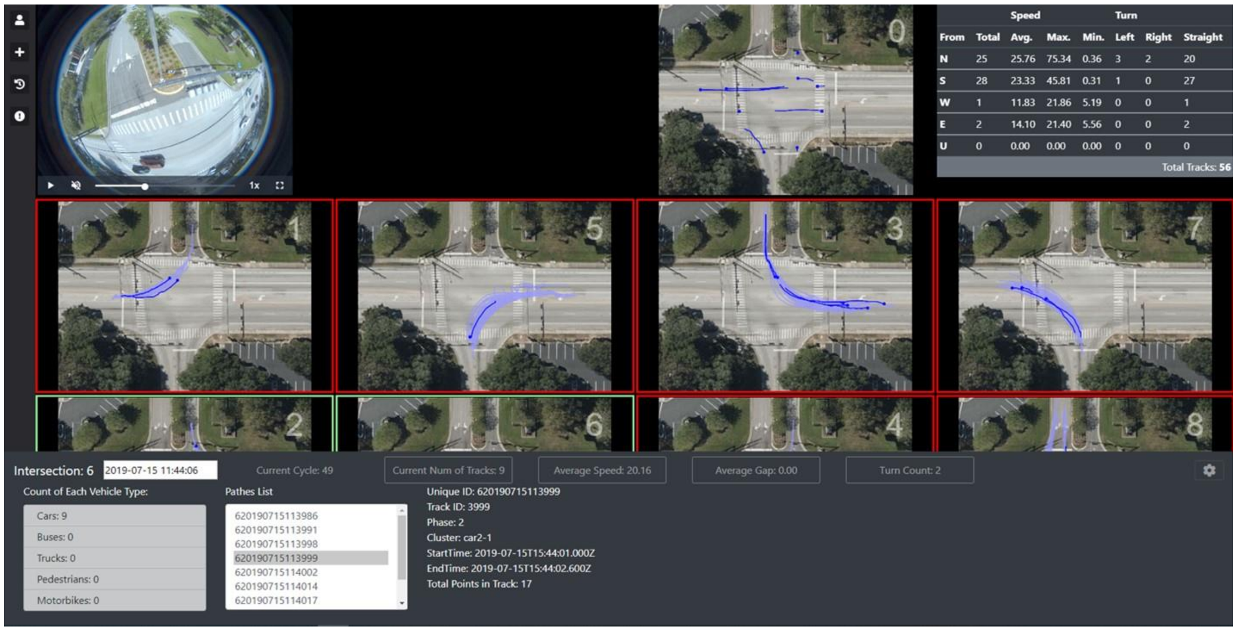
		Speed			Turn		
From	Total	Avg.	Max.	Min.	Left	Right	Straight
N	20	22.37	51.08	2.99	1	1	18
S	8	17.03	32.90	0.25	0	0	8
W	1	11.17	15.02	5.10	0	0	1
E	1	11.73	17.83	4.59	0	0	1
U	0	0.00	0.00	0.00	0	0	0

Total Tracks: 30

Intersection: 6 2019-07-15 11:11:37 Current Cycle: 35 Current Num of Tracks: 3 Average Speed: 16.33 Average Gap: 0.00 Turn Count: 0

Phases Arrangement: 1 5 3 7 / 2 6 4 8
Turn Count
Scaling of intersection: 1
Historical Length(Seconds): 120
Display Length(Seconds): 3

Figure 5-12. The settings menu. It offers several options to the users for enhanced display.



		Speed			Turn		
From	Total	Avg.	Max.	Min.	Left	Right	Straight
N	25	25.76	75.34	0.36	3	2	20
S	28	23.33	45.81	0.31	1	0	27
W	1	11.83	21.86	5.19	0	0	1
E	2	14.10	21.40	5.56	0	0	2
U	0	0.00	0.00	0.00	0	0	0

Total Tracks: 56

Intersection: 6 2019-07-15 11:44:06

Current Cycle: 49

Current Num of Tracks: 9

Average Speed: 20.16

Average Gap: 0.00

Turn Count: 2

Count of Each Vehicle Type:

- Cars: 9
- Buses: 0
- Trucks: 0
- Pedestrians: 0
- Motorbikes: 0

Pathes List

- 620190715113986
- 620190715113991
- 620190715113998
- 620190715113999
- 620190715114002
- 620190715114014
- 620190715114017

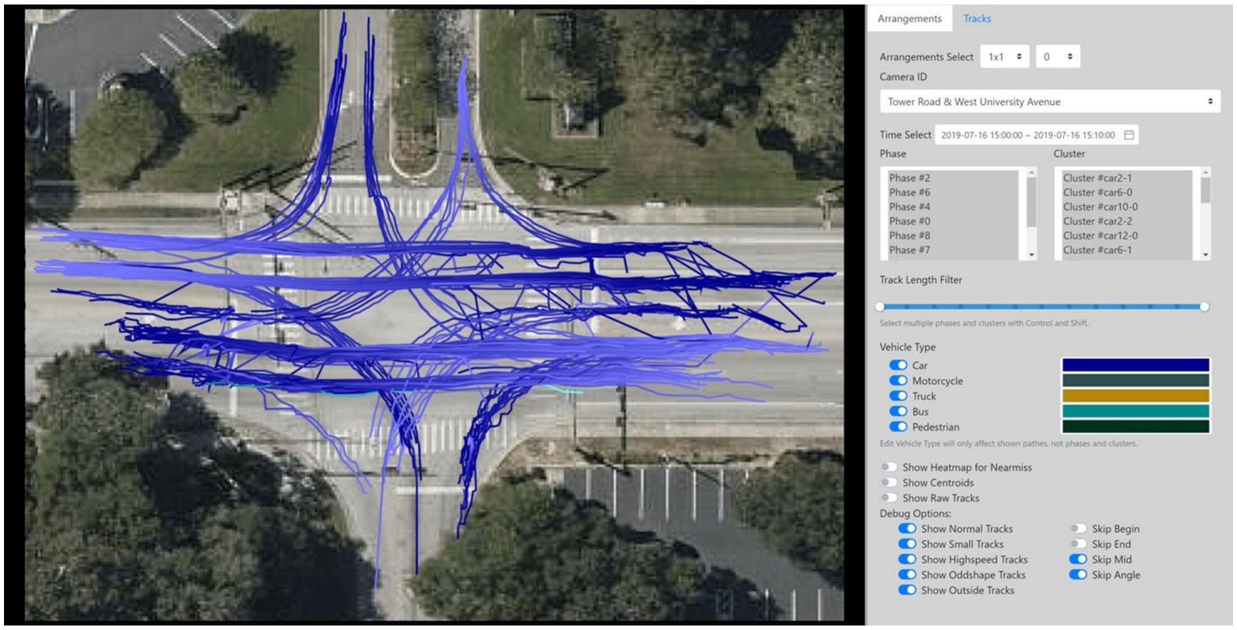
Unique ID: 620190715113999
Track ID: 3999
Phase: 2
Cluster: car2-1
StartTime: 2019-07-15T15:44:01.000Z
EndTime: 2019-07-15T15:44:02.600Z
Total Points in Track: 17

Figure 5-13. To get the details of an ongoing trajectory, click on settings and then the button Current Num of Tracks.

5.4. Historical Analysis

Our visualization software may be used for mining the trajectories created in the past from traffic videos. In this section, we describe the features supported for historical analysis. Figure 5-14 shows the landing page for historical analysis. It provides the users with capabilities to filter the trajectories based on time of occurrence, phase, and cluster.

Further, the user may choose to view the cluster centroids of the dominating clusters, or the anomalous trajectories, or even filter out the trajectories impacted by noises. It also supports the side-by-side display of trajectories for different combinations of time of the day, or day for the week, or classes of objects. We present some of the possibilities in this section.


Arrangements | Tracks

Arrangements Select: 1x1, 0

Camera ID: Tower Road & West University Avenue

Time Select: 2019-07-16 15:00:00 ~ 2019-07-16 15:10:00

Phase	Cluster
Phase #2	Cluster #car2-1
Phase #6	Cluster #car6-0
Phase #4	Cluster #car10-0
Phase #0	Cluster #car2-2
Phase #8	Cluster #car12-0
Phase #7	Cluster #car6-1

Track Length Filter

Select multiple phases and clusters with Control and Shift.

Vehicle Type
- [✓] Car
- [✓] Motorcycle
- [✓] Truck
- [✓] Bus
- [✓] Pedestrian

Edit Vehicle Type will only affect shown pathes, not phases and clusters.

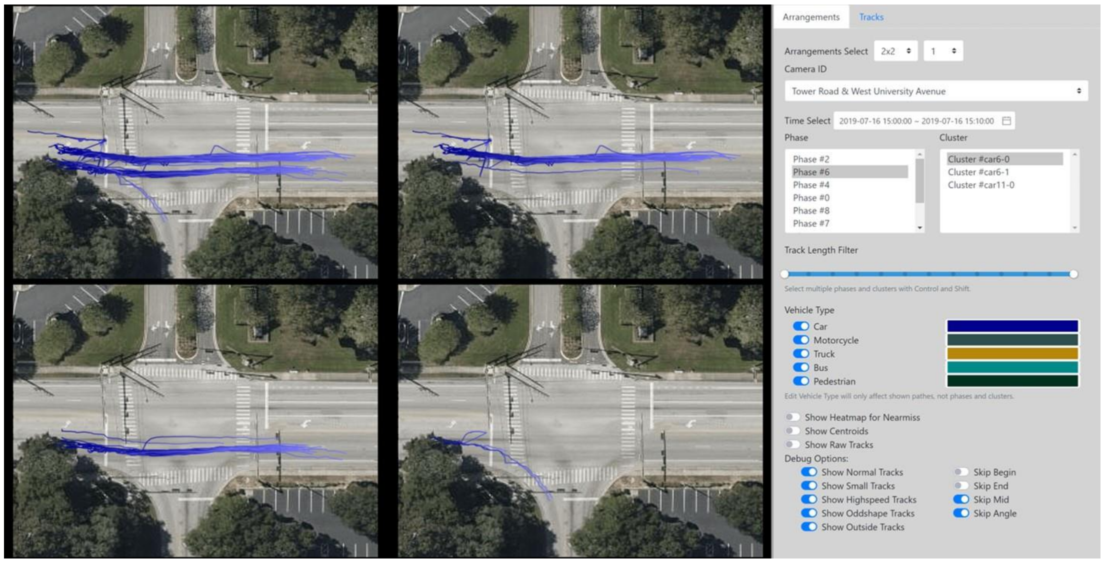
- [ ] Show Heatmap for Nearmiss
- [ ] Show Centroids
- [ ] Show Raw Tracks

Debug Options:

[✓] Show Normal Tracks	[ ] Skip Begin
[✓] Show Small Tracks	[ ] Skip End
[✓] Show Highspeed Tracks	[✓] Skip Mid
[✓] Show Oddshape Tracks	[✓] Skip Angle
[✓] Show Outside Tracks	

Figure 5-14. The display for historical analysis. It provides several options for filtering a large collection of trajectories. The primary filter is based on the time range. All tracks that happened for 10 minutes starting at Monday, July 15, 2019, 15:00:12 EDT are displayed here.

Selecting a Time Window for Trajectories – The time slider, used for selecting a time window, is the primary filter. It is a range slider with a flexible start and end. Once a time window is chosen, all the trajectories occurring in that time window on the selected intersection are obtained using a database query and displayed on the screen. The trajectories in Figure 5-14 show such an example, where the length of the time window is 10 minutes starting at Monday, July 15, 2019, 15:00:12 EDT.



Arrangements Select: 2x2 · 1
Camera ID: Tower Road & West University Avenue
Time Select: 2019-07-16 15:00:00 ~ 2019-07-16 15:10:00

Phase: Phase #2, [✓ Phase #6], Phase #4, Phase #0, Phase #8, Phase #7
Cluster: [✓ Cluster #car6-0], Cluster #car6-1, Cluster #car11-0

Track Length Filter

Select multiple phases and clusters with Control and Shift.

Vehicle Type

[✓ Car]	🟦
[✓ Motorcycle]	⬛
[✓ Truck]	🟨
[✓ Bus]	🟩
[✓ Pedestrian]	⬛

Edit Vehicle Type will only affect shown pathes, not phases and clusters.

Show Heatmap for Nearmiss
Show Centroids
Show Raw Tracks

Debug Options:

[✓ Show Normal Tracks]	Skip Begin
[✓ Show Small Tracks]	Skip End
[✓ Show Highspeed Tracks]	[✓ Skip Mid]
[✓ Show Oddshape Tracks]	[✓ Skip Angle]
[✓ Show Outside Tracks]	

Figure 5-15. A combination of phases and clusters were used to display all trajectories in phase 6 and their respective clusters. These clusters automatically partition the trajectories based on which lane they occur and between the through and right turn trajectories.

Selecting Phases and Clusters – All trajectories in a given time window may be further partitioned based on their movement, and/or cluster. Figure 5-15 showcases a selection combination for phases and clusters. The trajectories on phase 6 here are partitioned into three clusters. Two of these, namely, ’car6-0’ and ’car6-1’, correspond to the trajectories on the two lanes, respectively, while ’car11-0’ corresponds to the right turn trajectories. These three clusters are shown in the three remaining windows. The remaining cluster ’small’ corresponds to the tiny tracks that were created due to processing errors and are not shown in this figure. Figure 5-15 demonstrates the powerful feature of our software, which enables a side-by-side comparison of sets of trajectories, where each set may be chosen independently using any criteria. The “Phase” window often has a phase 0, which is simply a phase that the trajectories are assigned when the software cannot determine their movement direction, which happens for anomalous trajectories and also for trajectories resulting from a processing error.

Cluster Centers – The visualization tool may be used to display the cluster centers used for the online cluster assignment. This is useful in sampling the major movements at an intersection. It also comes in handy in debugging mismatched alignment and stitching small tracks. Figure 5-16 shows the cluster centers corresponding to the trajectories of the 10 minutes duration chosen earlier.

Heat map for Near-misses – Figure 5-17 shows how heat maps would appear to highlight areas of the intersection that are unsafe and have had more near-misses. The heatmap in this figure is based on a hypothetical dataset. Two near-miss scenarios are defined for videos, namely, a spatial scenario where road objects collide or are very close in image space, and temporal scenario where a dramatic speed change is observed to avoid near-misses (e.g. a sudden brake). We use distance-based measures and speed measures to compute the near-miss probability of road objects with the average of two scores serving as the final output as described below.

- Spatial distance measure: We use tracks data to form trajectories of road objects and compute distances between two road objects using center coordinates of detected bounding boxes in image space at the frame level. The probability of a spatial near-miss situation is computed using the Euclidean distance of road objects with ratio to the size of object according to object class (vehicles size of each class does not vary much).

- Temporal motion measure: The speed of the road object is computed by adjacent displacement over multiple time frames. The probability of motion-based near-miss is computed by the fractional decrease in speed.

We use a weighted average of the above two probabilities of near-miss to compute the overall score. We used different spatial and temporal thresholds for different intersections, and these are parameters to our algorithms. The thresholds may be set to different values for different intersections.

Our current near-miss detection algorithm is a descriptive rather than a predictive method. So, it is meant for offline analysis where we look at frames before and after the reference frame to detect a near-miss situation. Developing a real-time predictive near-miss algorithm needs a completely different framework. For applying the algorithm in real-time we would need the following:

- Hardware: fast GPU for video processing and fast CPU for near-miss detection
- Software: batch processes have to be converted to online processes (so we will not have to use future frame information).

The real-time solution must have good scalability so that it may be applied to (1) other data (non-fisheye data or new fisheye intersection) and (2) other related tasks (e.g., heat map generation or near-miss prediction).

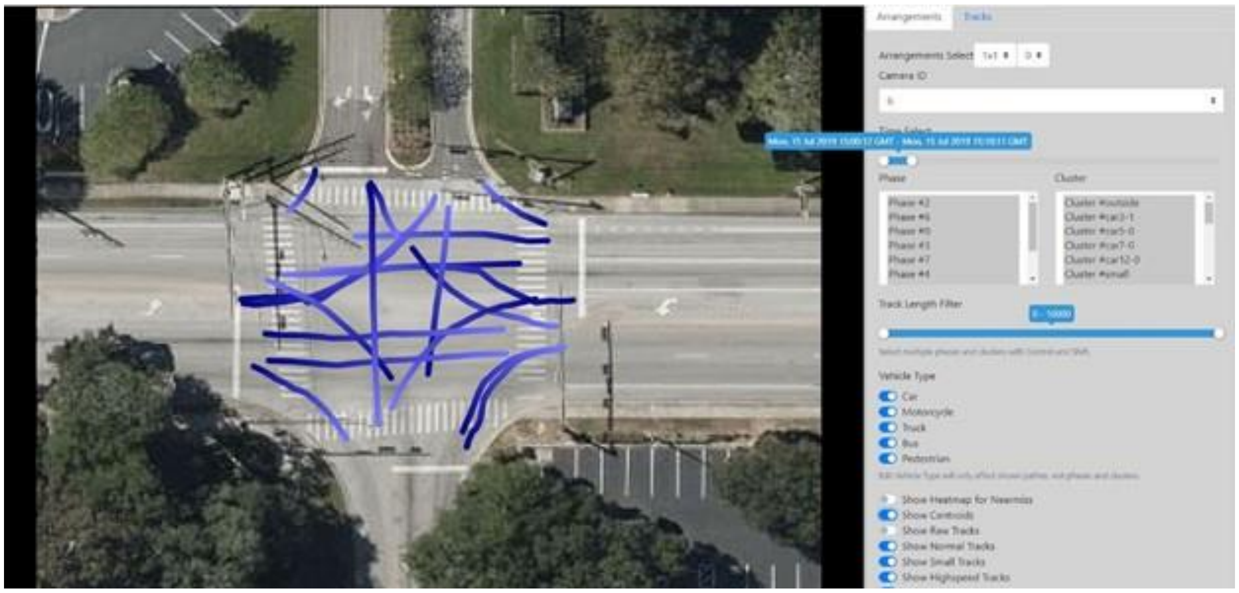


Figure 5-16. The centroids or the cluster centers of the trajectories

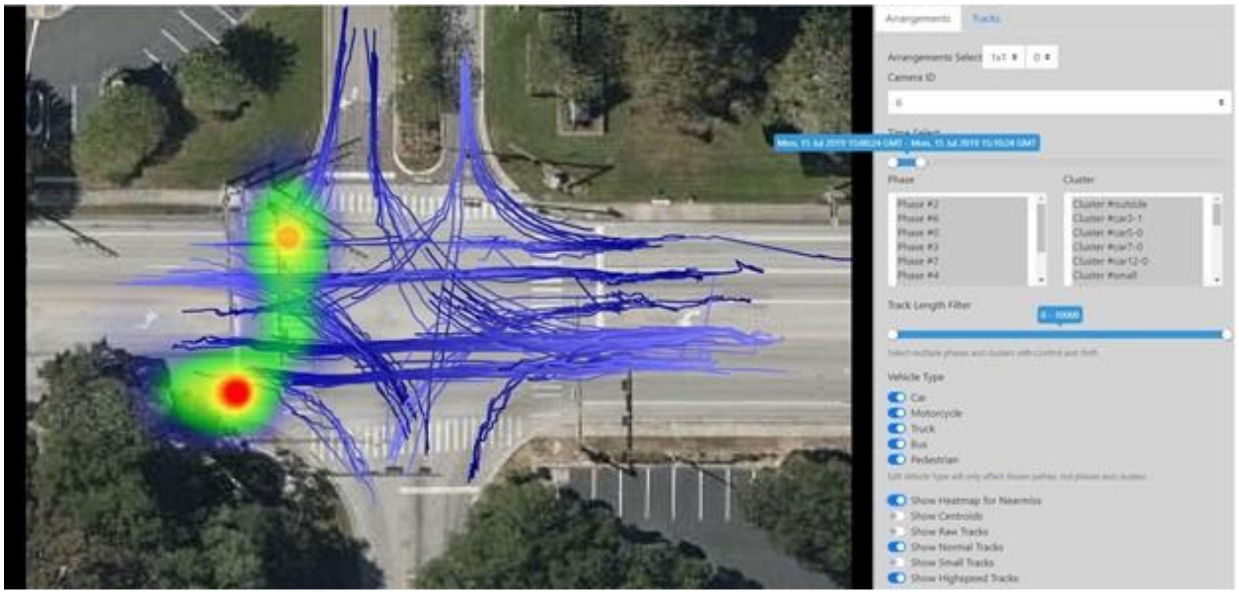
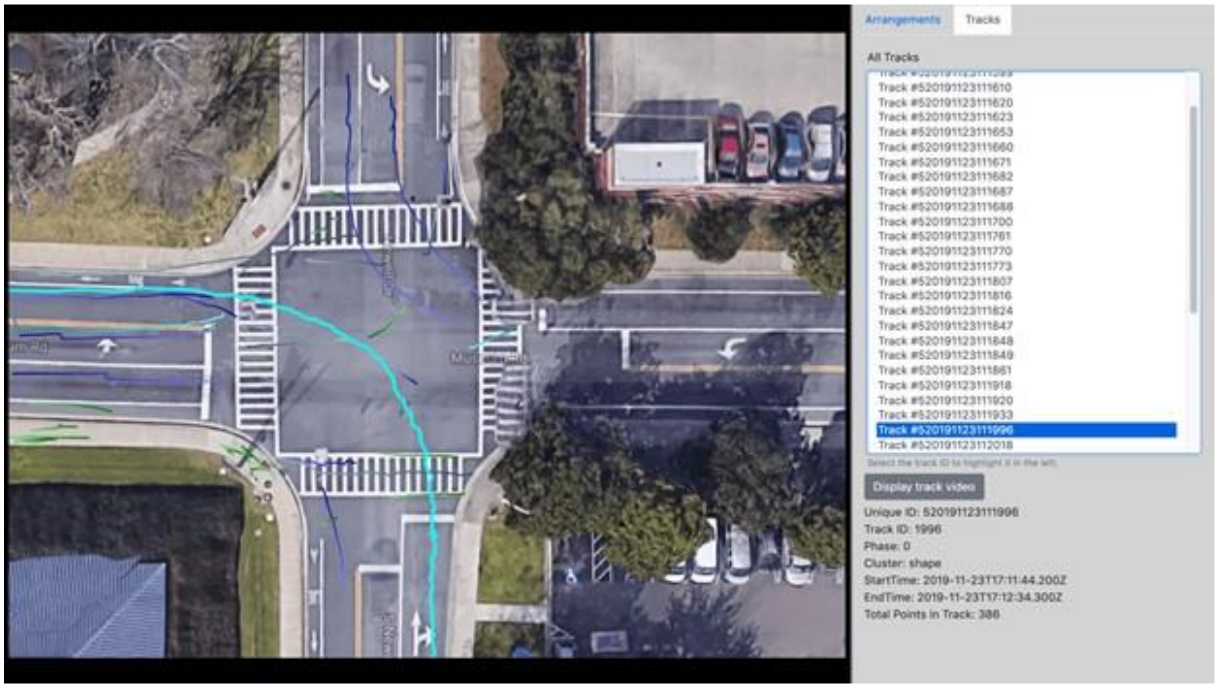


Figure 5-17. A heatmap based on hypothetical near-miss or crash events.

Anomalous Tracks – A track may be anomalous for several reasons such as its shape does not conform to any of the cluster centroids, or when it occurs while the ongoing signal is red. In Figure 5-18, we have highlighted an anomalous trajectory in which a left turn was made from a through lane, even though there is a left-turn lane at this intersection.



Unique ID: 520191123111996
Track ID: 1996
Phase: 0
Cluster: shape
StartTime: 2019-11-23T17:11:44.200Z
EndTime: 2019-11-23T17:12:34.300Z
Total Points in Track: 386

Figure 5-18. All trajectories that are identified as anomalous. The one highlighted takes a left turn from a through lane.

5.5. Case Study: Traffic Trend Analysis

We present some results for trend analysis using the tiling and the rest of the system described earlier. Figure 5-19 shows the time of day and day of the week trends for the main clusters of car trajectories, the figures in each column are from a different day, and the figures in each row are for a different time.

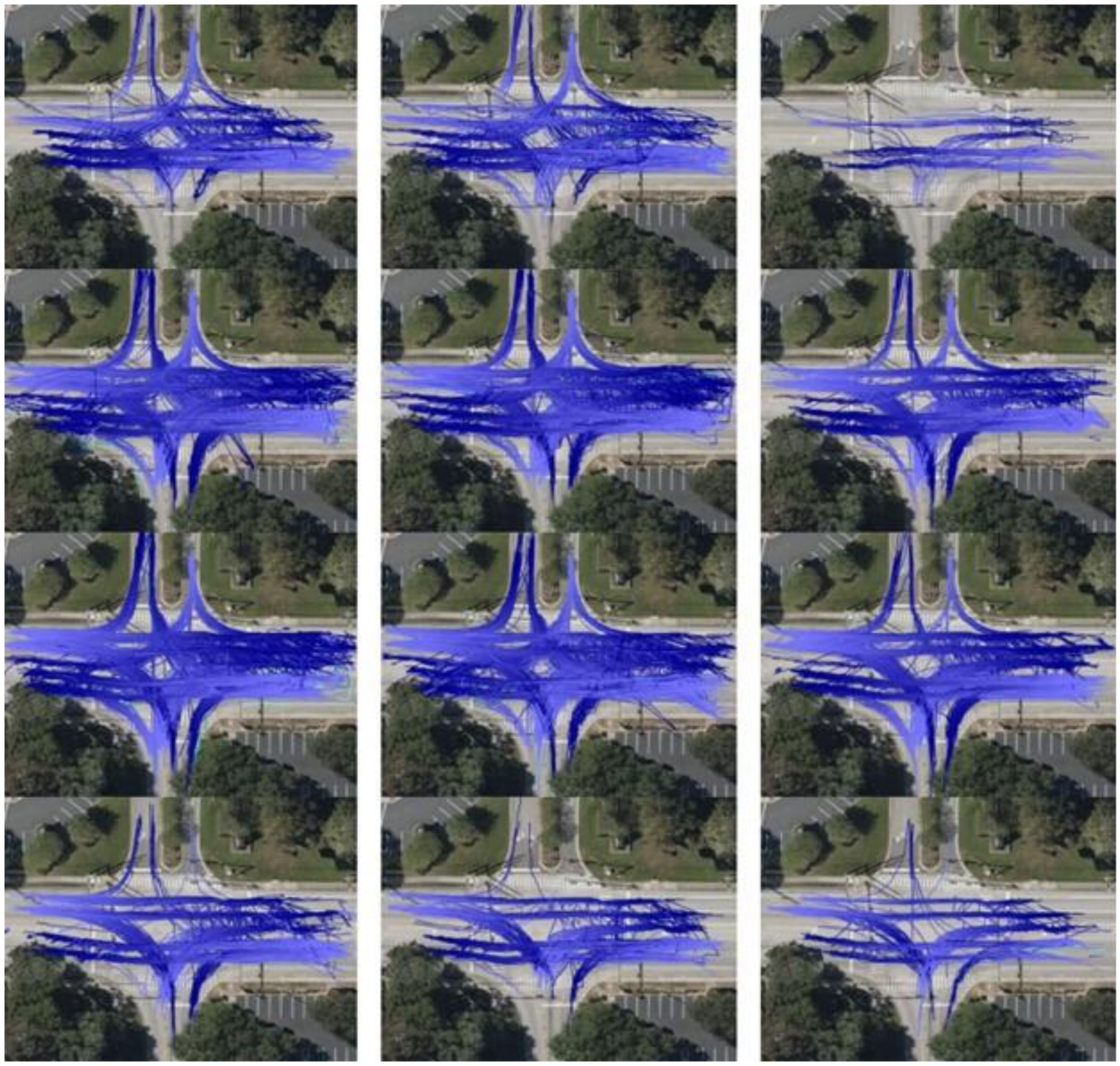
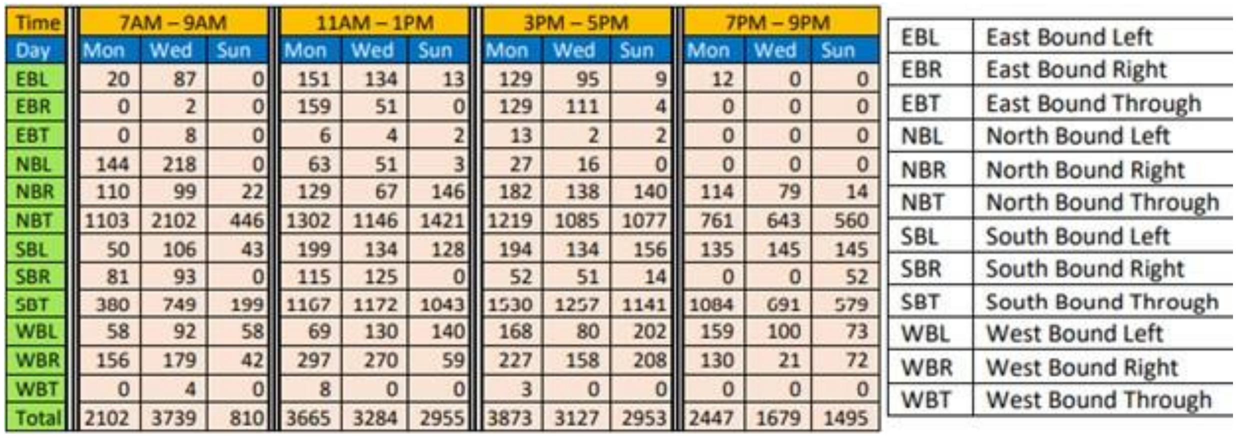


Figure 5-19. Clusters for trend analysis. Each column is for a different day. The left, middle, and right columns are for Monday, Wednesday, and Sunday, respectively. Each row is for a different time of day. The top row is early morning (7 AM–9 AM), the second row is late morning (11 AM–1 PM), the third row is afternoon (3 PM–5 PM), and the last row is late evening (7 PM–9 PM). There are fewer cars on Sundays. The number of blue tracks starting from the left road on the top becomes very few outside of the working hours and more during working hours, suggesting the presence of an office complex near the top of the image. Monday sees a late start and a late end of the day, and there is more northbound traffic in the morning and southbound traffic in the evening, suggesting the presence of a residential area in the south.

Specifically, the figures on the left, middle, and right columns are from a Monday, Wednesday, and Sunday, respectively. The figures on each row are from different two-hour intervals during the early morning (7 AM–9 AM, top row), mid-morning (11 AM–1 PM, second row from the top), afternoon (3 PM–5 PM, third row from the top), and evening (7 PM–9 PM, the last row). Figure 5-20 presents the corresponding exact count of vehicles per direction (east, west, north, south) and movement (through, right, left).

From Figure 5-19, as well as from the table in Figure 5-20, we observe the following:

- A significant reduction in the volume of cars occurs on Sunday, especially during the early morning and late evening hours, compared to that on a weekday. For example, there were about 60% and 40% fewer cars during the intervals 7 AM–9 AM and 7 PM–9 PM, respectively, on a Sunday compared to Monday.
- There are fewer cars during the mid-morning and after-noon hours on a Sunday as compared to similar times on Monday (by around 20%).
- The orange egress tracks (EBL) and the corresponding blue ingress tracks (NBL) near the middle of the peak of the image during working hours. Thus, we infer that the top of the image is an office complex. The ingress and egress traffic patterns reveal a potential optimization of the signal plan by allowing the most green time to the major through movements (NBT and SBT) during the non-working hours.
- The total volume of cars during the early morning hours of Monday is fewer than that of Wednesday by about 50%, while there are about 40% more cars in the late evening on Monday than on Wednesday, which indicates a late start and a late end of the day on Monday, the first day of the workweek.
- For this particular intersection, more cars are northbound during the morning and more cars that are southbound during the evening, which probably indicates that the major residential areas nearby are towards the south.



Time	7AM – 9AM			11AM – 1PM			3PM – 5PM			7PM – 9PM		
Day	Mon	Wed	Sun	Mon	Wed	Sun	Mon	Wed	Sun	Mon	Wed	Sun
EBL	20	87	0	151	134	13	129	95	9	12	0	0
EBR	0	2	0	159	51	0	129	111	4	0	0	0
EBT	0	8	0	6	4	2	13	2	2	0	0	0
NBL	144	218	0	63	51	3	27	16	0	0	0	0
NBR	110	99	22	129	67	146	182	138	140	114	79	14
NBT	1103	2102	446	1302	1146	1421	1219	1085	1077	761	643	560
SBL	50	106	43	199	134	128	194	134	156	135	145	145
SBR	81	93	0	115	125	0	52	51	14	0	0	52
SBT	380	749	199	1167	1172	1043	1530	1257	1141	1084	691	579
WBL	58	92	58	69	130	140	168	80	202	159	100	73
WBR	156	179	42	297	270	59	227	158	208	130	21	72
WBT	0	4	0	8	0	0	3	0	0	0	0	0
Total	2102	3739	810	3665	3284	2955	3873	3127	2953	2447	1679	1495

EBL	East Bound Left
EBR	East Bound Right
EBT	East Bound Through
NBL	North Bound Left
NBR	North Bound Right
NBT	North Bound Through
SBL	South Bound Left
SBR	South Bound Right
SBT	South Bound Through
WBL	West Bound Left
WBR	West Bound Right
WBT	West Bound Through

Figure 5-20. Turning and through movement counts for various time periods in a single day.

5.6. Conclusions

We have developed a novel visual analytics software that is designed to serve the broader user community of traffic practitioners in analyzing the efficiency and safety of an intersection. It uses data generated from video analysis that corresponds to spatial locations and timestamps of all the objects passing through an intersection along with their object type.

The tool has two modes of operations, streaming mode and historical. Using both of these modes as appropriate, a traffic engineer can quickly analyze the normal clusters and anomalous clusters to get a better understanding of traffic behavior at an intersection. Our software helps diagnose the following type of scenarios:

- It can help detect when a large fraction of red-light violations occurs at an intersection. Once a problem area is detected, the traffic practitioner can potentially address this by suitably address signal timing plans for an intersection or neighboring intersections on a corridor.
- High traffic volume on a turning movement, but comparatively lower green time assigned to it. This traffic pattern may vary by the time of day or the day of the week.

The tool provides a simple interface for filtering and selection mechanisms for the desired trajectories, which can be based on time of day, day of the week, object type (pedestrian, cyclist, car), and signal time phase. We have demonstrated the use of our tool to filter trajectories to study them further and applied the software on an intersection to show clustering results and examples of anomalous trajectories. Also, the software could be useful in studying trends in traffic patterns for different times of the day and different days of the week.

Chapter 6 – Develop Algorithms and Software for Anomalous Trajectories

6.1. Introduction

The advent of nominally priced video-based systems, open-source tools for video processing and deep learning, and the availability of low-cost graphics processing units (GPUs) have opened the door for their use in real-time transportation decision systems. While video-based systems for intersection traffic measurement can perform multiple object detection and tracking, their use for more complex tasks such as anomaly detection and near-misses is limited. The recent proposed AI city challenge (Tang et al., 2019) also focuses on similar applications. In general, monitoring activities of road users and understanding traffic events have shown to be useful for modeling, analyzing, and improving road-based transportation.

An anomalous trajectory is a trajectory or part of a trajectory that is significantly different from others in a cluster of trajectories in terms of either shape or velocity. Detection of anomalous trajectories is important as these are often at the center of near-miss or other hazardous events. Examining the trajectory of the vehicles in relation to the signal state can also help quantify more accurately the start-up lost time (time between the signal turning green and the vehicle stream starting to move), the saturation headways, gap acceptance behavior, or, in more extreme cases, vehicles jumping a red signal. After identifying the anomalous tracks, it is possible to analyze and categorize these tracks to further study the prominent behavioral tendencies among drivers and pedestrians at an intersection, with a goal to improve its design and operations such that they minimize or discourage such tendencies. We have discussed the anomalous vehicle trajectories in Section 7.3.

The focus of the current report is to build upon the video processing techniques developed as part of the prior tasks and implement advanced analysis algorithms for identifying anomalous trajectories for pedestrians and vehicles. Specifically, we will be focusing on the following metrics:

1. A pedestrian jaywalks at an intersection, i.e., they do not follow the normal path followed by other pedestrians.
2. A vehicle stops in the middle of an intersection.
3. A vehicle sits on the stop bar for five or more seconds.
4. Number of jaywalks and the impact on the overall traffic and near-misses
5. Number of erratic vehicle behaviors and the number of traffic disruptions caused
6. Number of vehicles that wait on the stop bar for a long time and startup time lost because of this behavior
7. Number of vehicles running a red light
8. Number of vehicles changing lanes
9. Number of vehicles going the wrong way
10. Number of pedestrians over different time periods.

We present the results from our pedestrian anomaly analysis module in Section 7.2, and vehicle anomaly analysis in Section 7.3, and finally, we conclude in Section 7.4. Our results in this report are based on the intersection of W University Ave at 13th St.

6.2. Pedestrian Anomaly Analysis

The most dangerous behavior a pedestrian can engage in is crossing into oncoming traffic. Therefore, tracking and preventing occurrences of jaywalking is our primary metric of concern for pedestrian safety within any given intersection. Our analysis will also look at secondary safety metrics such as how many pedestrians cross on a given day, how long a pedestrian must wait to cross the intersection, if the pedestrian follows an abnormal path while doing so, and if they begin their crossing too early or finish it too late. These secondary metrics are then aggregated such that we may gain a better understanding as to the causes of jaywalking and help the city arrive at policies that have a chance of reducing it.

Pedestrian tracking data are first matched with signaling information obtained from high resolution controller logs provided by the county to determine if a pedestrian is crossing or waiting to cross. We define jaywalking as any pedestrian crossing that occurs completely outside of any signal cycle and a poor crossing as a crossing where only a portion of the trajectory is outside of the signal cycle. Once the jaywalking pedestrians are identified, they are aggregated temporally to search for patterns in jaywalking by phase, intersection, time of day, day of week, and so on. These aggregations can also be averaged by week to determine the average jaywalking on a Monday, for example, during a specific month of the year. This process is also repeated for all safety metrics throughout this analysis. The figures in this chapter are for November 16–23, 2020, on the West University Ave at SW 13th Street intersection.

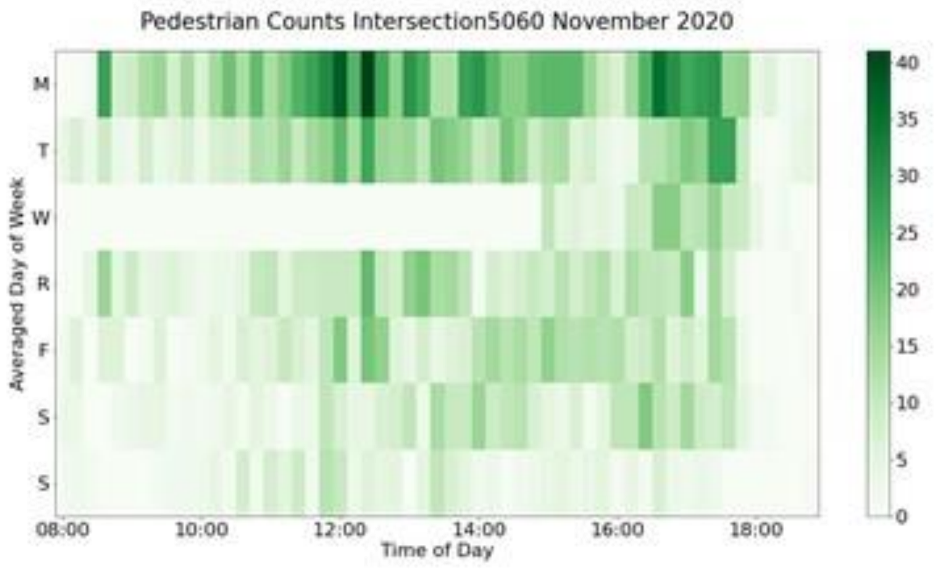


Figure 6-1. Each rectangular window represents 12 minutes of time as does every following chart; the color represents the number of pedestrians counted within that interval.

Due to network issues, much of Wednesday's videos were not able to be collected and are missing for Figure 6-1 and following ones as well. Figure 6-2 shows the pedestrian counts by the hour of the day averaged over all days of the week being studied, while Figures 6-3, 6-4, and 6-5 show various statistics on the jaywalkers.

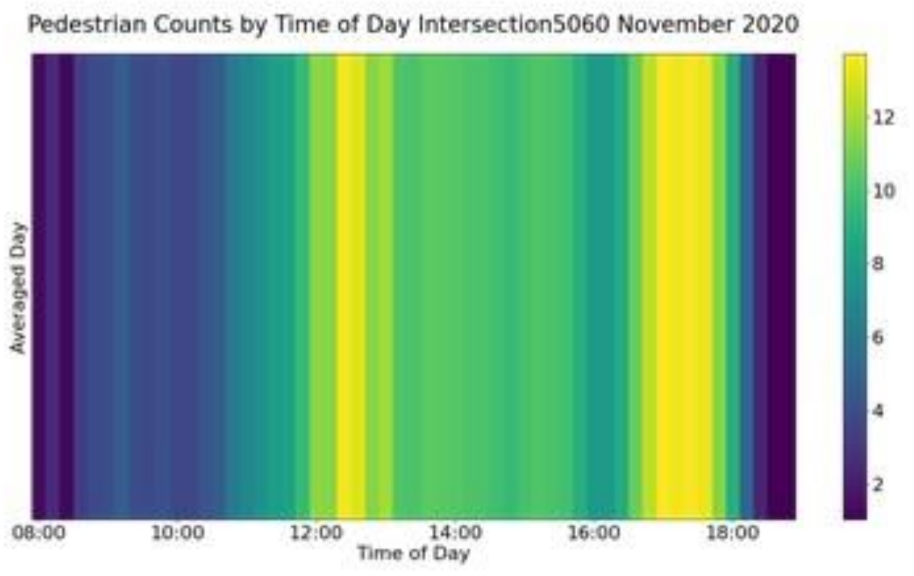


Figure 6-2. Each vertical rectangle represents a 12-minute window where the color represents the average number of pedestrians for that time of day in a given month.

An ongoing trend for many of our traffic metrics (pedestrian counts, jaywalking, waiting times, etc.) is a peak on Monday and a decline as the week goes on. However, we are likely missing alternative traffic patterns after daylight hours when we are no longer able to capture video. Pedestrian as well as vehicle behaviors would likely be affected by factors such as nighttime events in the midtown and downtown areas, a reduction in light, altered demographics (out of town traffic, ride sharing, etc.) as well as other factors.

Our ability to count jaywalkers is dependent on two things: signal information and the ability to track pedestrians relative to the crosswalk stop bar. Once a pedestrian's phase of movement is determined, the distance they are from the stop bar is calculated throughout their trajectory. Because each intersection and crosswalk are unique, it is also necessary to know the distance between the first and second stop bar to know which side of the crosswalk they are on. Pedestrians both outside the signal period and stop bar length (jaywalkers) are shown here:

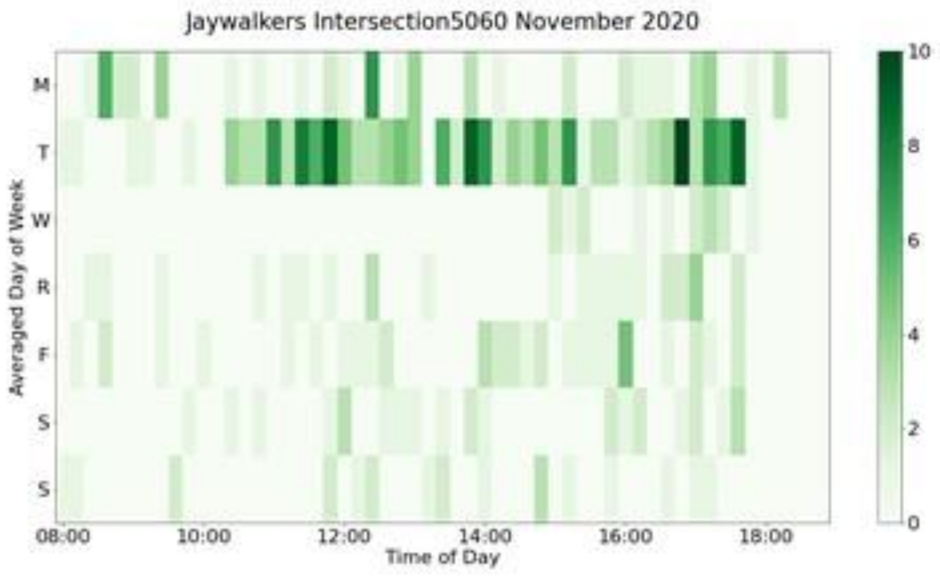


Figure 6-3. Color values represent the number of jaywalkers counted within that interval.

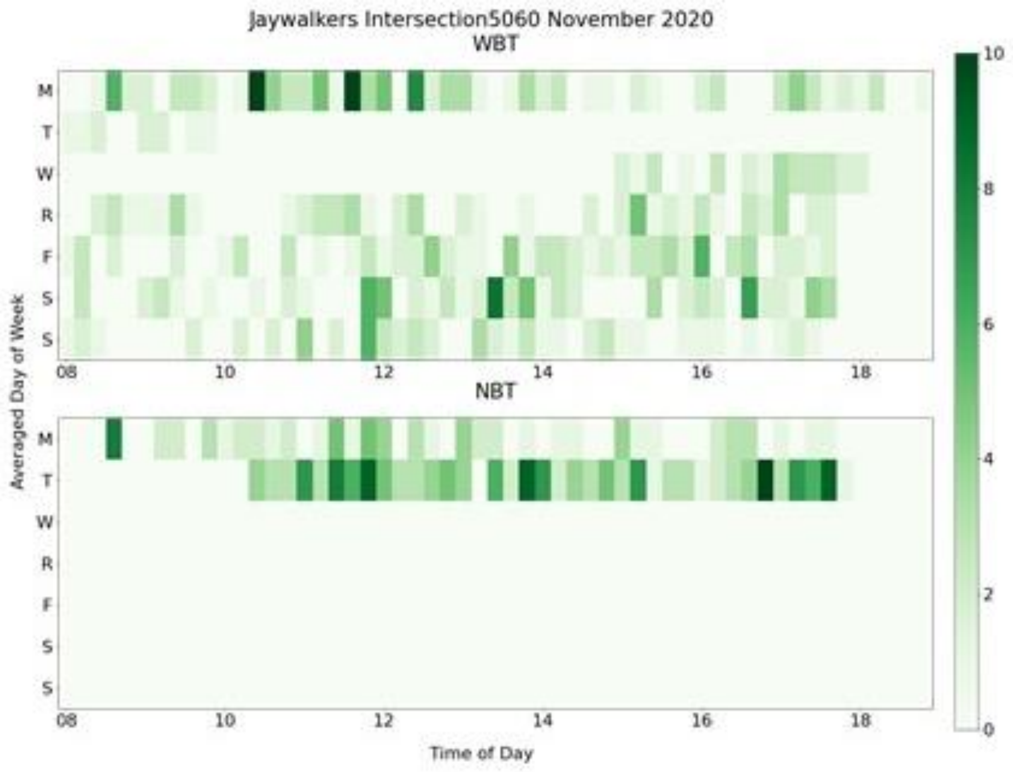


Figure 6-4. Color values represent the number of jaywalkers counted within that interval for their respective phases.

The observations regarding jaywalking are as follows:

- The Tuesday and Wednesday of this particular week were anomalies themselves due to abnormal traffic flow and signal timing.
- The time of day jaywalking most occurred was between 12:00 PM and 5:00 PM (after removing Tuesday and Wednesday).
- Monday had the most instances of jaywalking (75 instances), which is logical, given that many are adjusting back to their work schedule. People may be more likely to be late to their obligations and thus more incentivized to cross inappropriately.

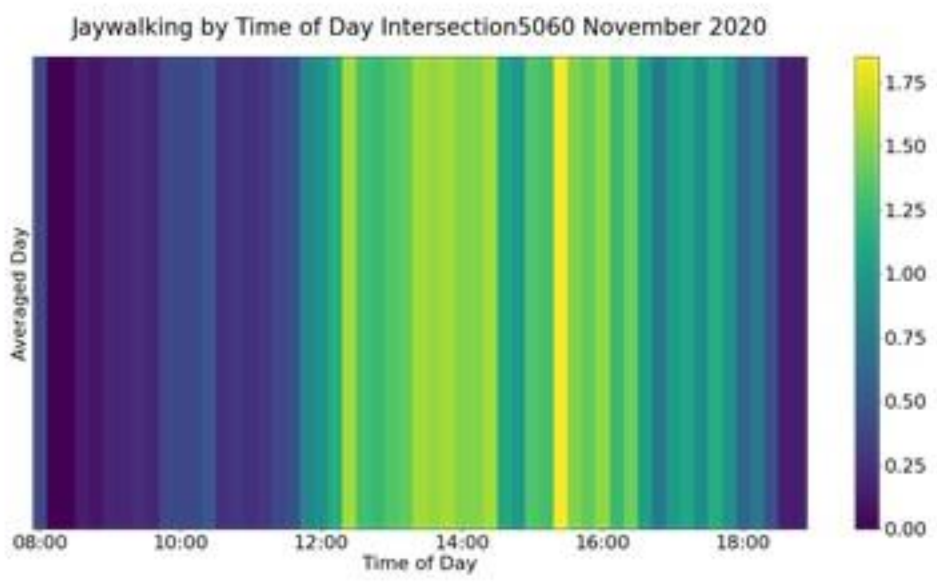


Figure 6-5. Each vertical rectangle represents the average number of jaywalkers for that time of day in a given month.

There are several reasons a pedestrian may jaywalk such as impatience, minimal traffic (a seemingly safe crossing opportunity), or other factors. A correlation between these factors can be evaluated once these variables are identified and sufficient data are gathered. Three possible correlates we measured were crosswalk waiting times, occurrences of early crossing, and occurrences of late crossing which are shown below. Figure 6-6 shows the crosswalk waiting times.

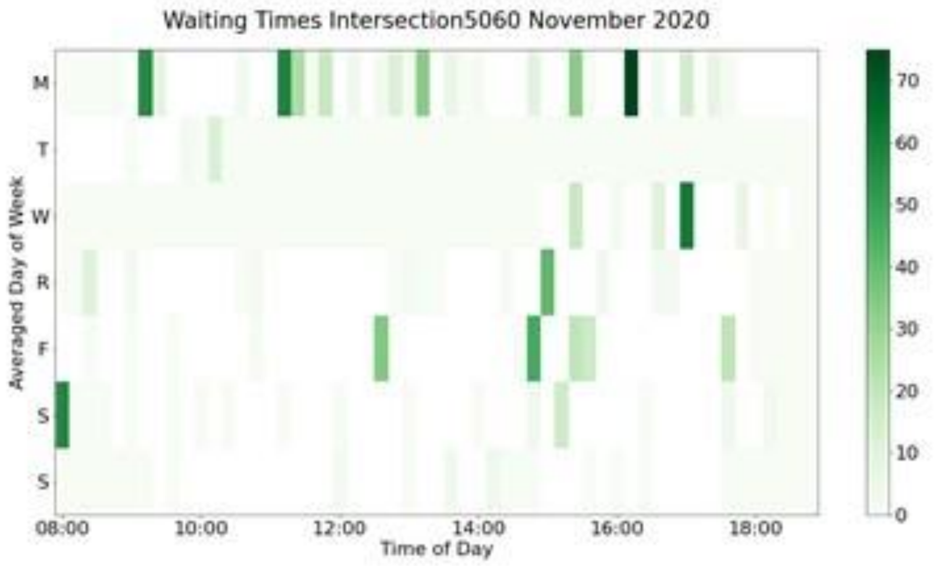
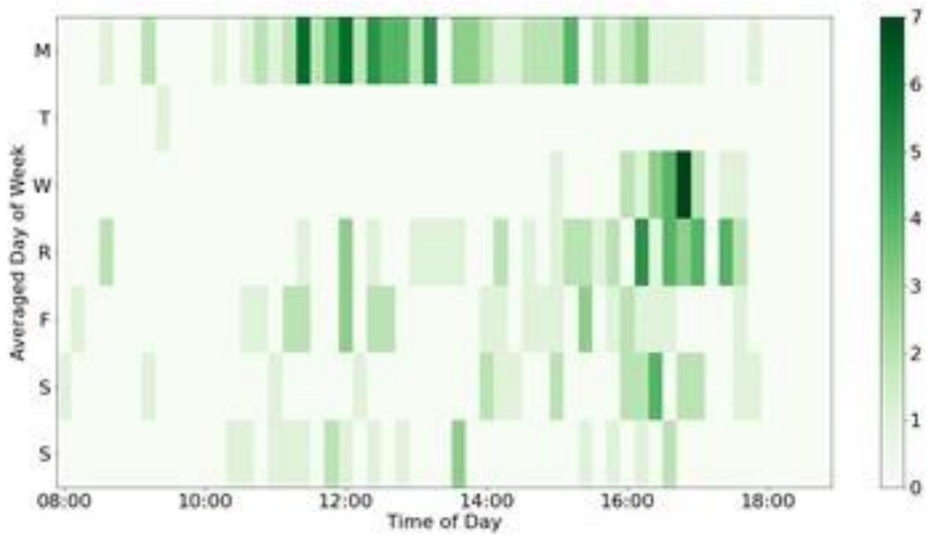


Figure 6-6. Each rectangular window represents 12 minutes of time; the color represents the average number of seconds needed to wait before an appropriate crossing is available in that interval.

- For waiting times it appears that signal times for pedestrians waiting to cross are well optimized overall except in certain windows of time where there are significantly longer wait times. This may be an artifact of video processing or simply that people who did not intend to cross were standing near the stop bar being marked as waiting. More data would be needed to reduce noise and give a clearer picture about typical waiting.

Figure 6-7 shows frequency of poor (early or late) crossings for time of day and day of week.



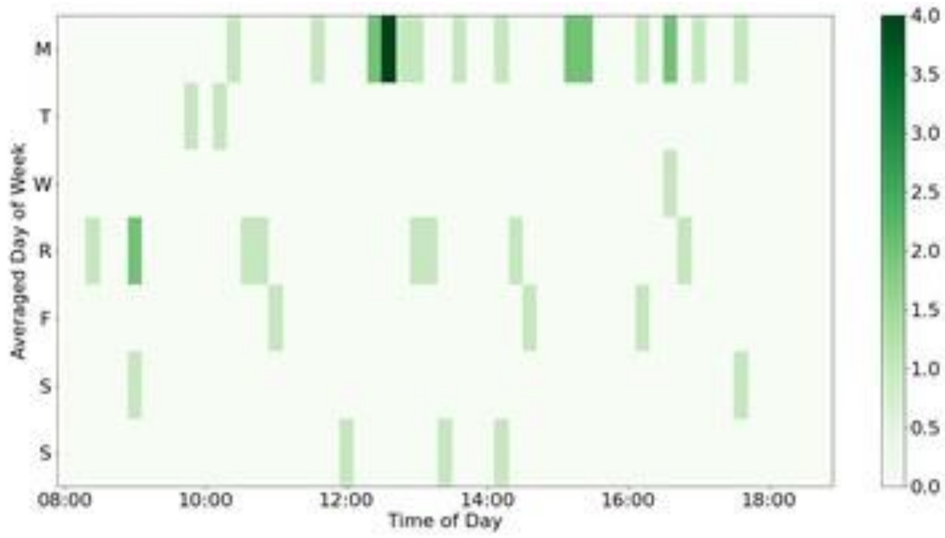


Figure 6-7. Each rectangular window represents 12 minutes of time; the color represents the number of poor crossings counted within that interval. Top: Insufficient crossing counts on Intersection 5060, Nov. 2020; bottom: Premature crossing counts on Intersection 5060, Nov. 2020.

Here we define insufficient crossing as a crossing that begins legitimately but ends outside of the signal cycle (late crossing), while premature begins illegitimately but ends within the signal cycle (early crossing). The ability to distinguish between late crossings that occurred because of a lack of crossing time allotment and those that occurred because of pedestrians arriving right before the signal period ended is being developed and will be available for the next report. While these charts show aggregations of behavior, we are also able to plot individual behavior, such as the crossing of pedestrians, frame by frame. Here the x axis represents time, and the y axis represents the pedestrian's distance from the crosswalk stop bar.

We present here a particular anomaly that was detected by our software.

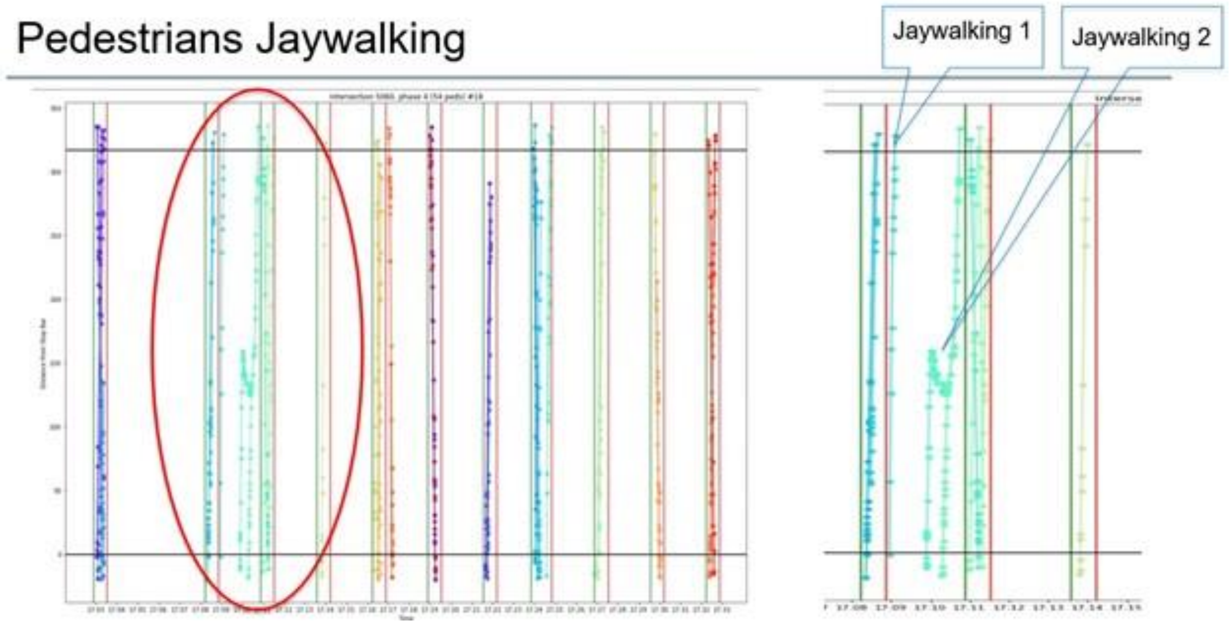


Figure 6-8. Pedestrian location in time and space. The horizontal axis represents the time, while the vertical axis presents the distance the pedestrians have travelled on the crosswalk. The vertical green and red lines correspond to "walk" and "don't walk" signal.

The jaywalking events in Figure 6-8 are shown in the fisheye camera views in Figures 6-9 and 6-10.

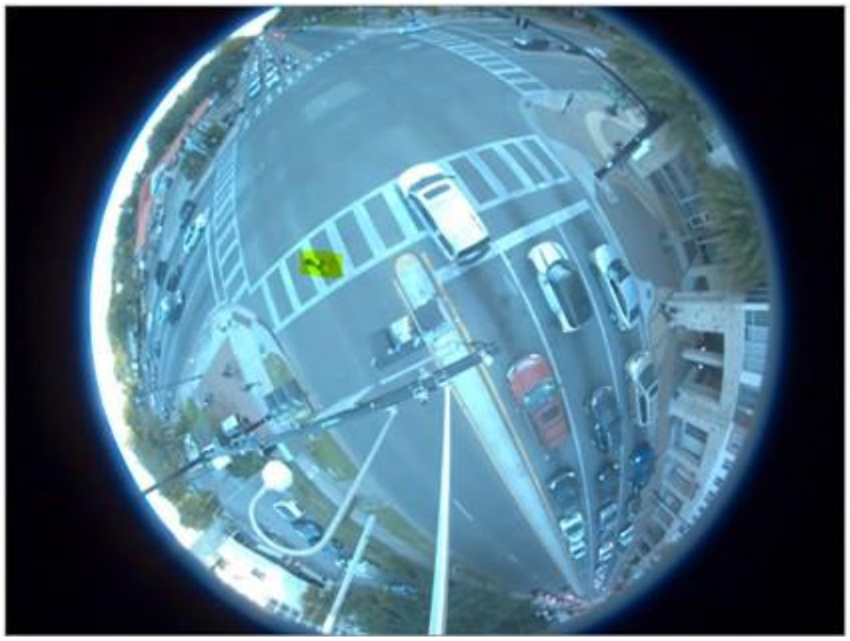


Figure 6-9. Jaywalking instance 1 found by our software

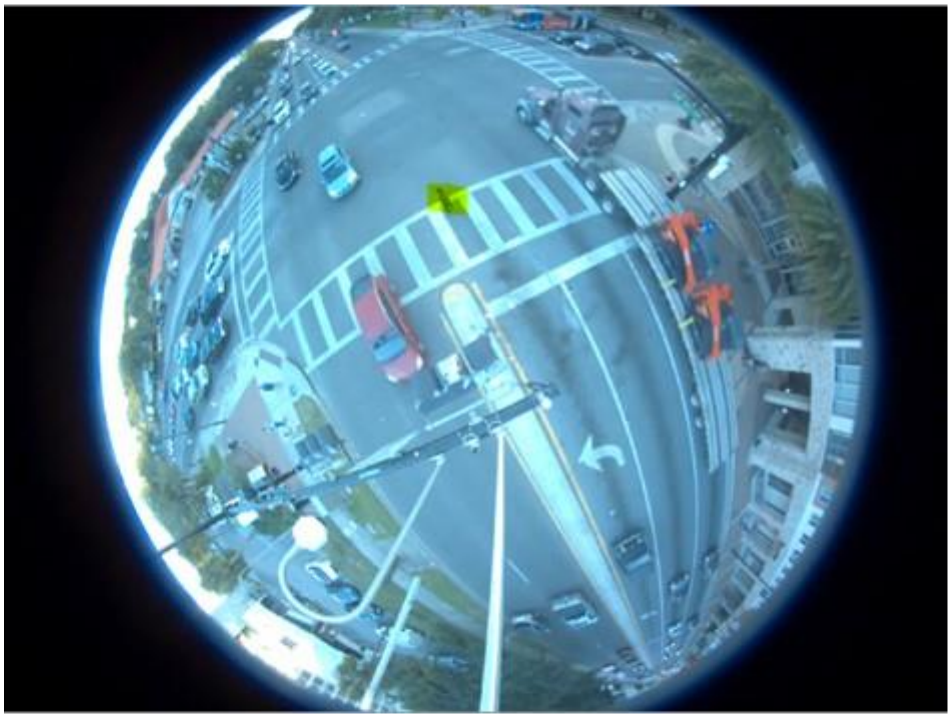


Figure 6-10. Jaywalking instance 2 found by our software

The effect of neighboring vehicles is prominent when we focus on the specific times these jaywalking events happened, which was between 17:10 and 17:12 on November 16, 2020. Figure 6-11 shows an analysis of the gap between vehicles during this specific time. However, if we aggregate the data, such effects are completely lost, unless there is a related incident.

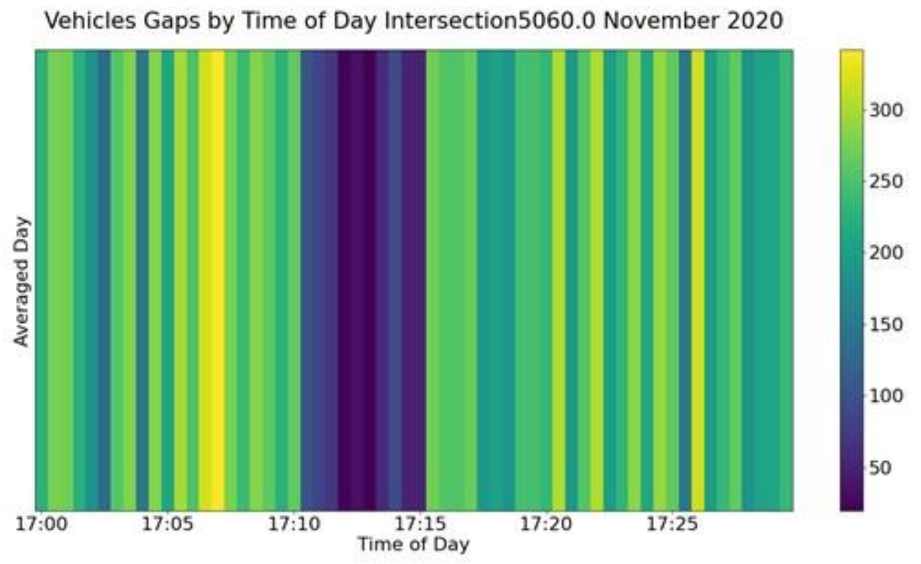


Figure 6-11. The gaps between vehicles measured in pixels. The gap reduces during the specific time when the jaywalking happens. and continues a few minutes later.

It is likely that vehicles who become aware of jaywalkers slow down to better react to the possibility of increasingly risky pedestrian behavior. This leads to other vehicles also slowing down in response, lowering the distance vehicles can safely move relative to one another. The aggregation of this behavior removes this phenomenon since larger time windows include vehicle who were not affected by this event, greatly skewing the average gap metric. Likewise, aggregation of other movement phases has a similar effect.

6.3. Vehicle Anomaly Analysis

For vehicles, there are a number of metrics we've looked at including vehicle counts, speeds, lane changes, red jumps, gaps between vehicles, wrong way movement, time to collision, and vehicle crash potential index. In the same manner as pedestrians, vehicles traveling through the intersection are tracked and classified as a car, truck, bus, or motorbike and then stored in the database for further analysis. The metrics mentioned above are shown in Figures 6-12 and 6-13.

We also evaluated a relatively new safety metric called CPI or Crash Potential Index. CPI is a measure of the safety (or lack thereof) between two vehicles or between a pedestrian and a vehicle. It is calculated as the sum of the probabilities of the maximum available deceleration (MADR) of a vehicle being lower than the deceleration needed to avoid a collision (DRAC) multiplied by the time window size and divided by the total time that vehicle was observed (Nadimi et al., 2016). Vehicle counts are shown here in Figures 6-12 and 6-13.

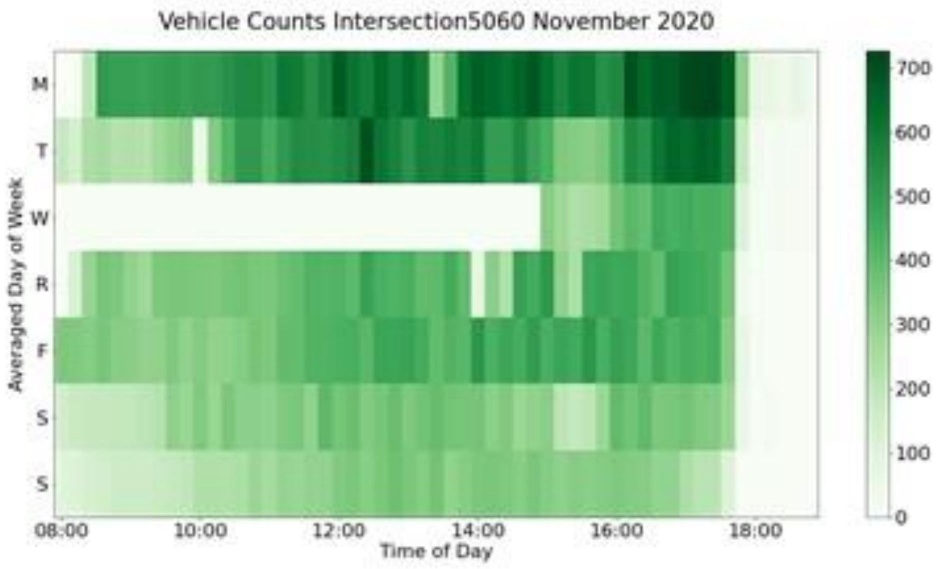


Figure 6-12. Each rectangular window represents 12 minutes of time; the color represents the number of vehicles counted within that interval.

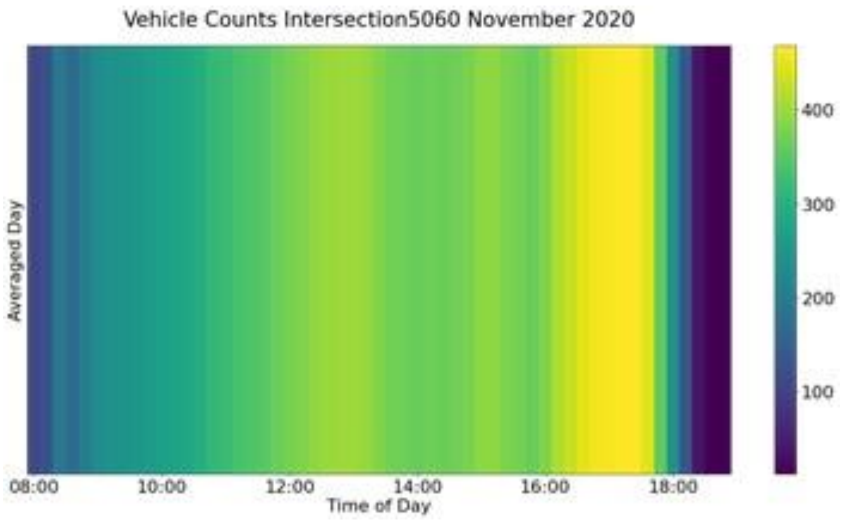


Figure 6-13. Each vertical rectangle represents the average number of vehicles for that time of day in a given month

Findings:

- Weekday traffic was most congested around 5 PM, with Monday being the most congested and gradually abating each weekday.
- Saturday saw traffic highest around 10 AM while Sunday peaked around 4 PM.
- Sunday is also the day with the lowest observed traffic.
- The Saturday traffic pattern is almost the exact opposite of Sunday, which suggests we should assess them individually in terms of traffic measurements and policy.

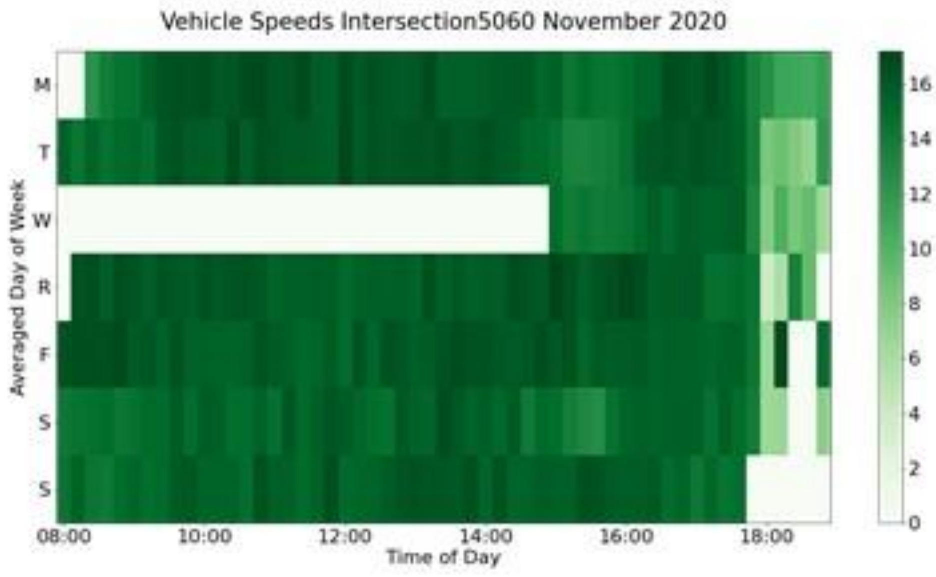


Figure 6-14. Each rectangular window represents 12 minutes of time; the color represents the number of vehicles counted within that interval with a speed of at least 5 mph.

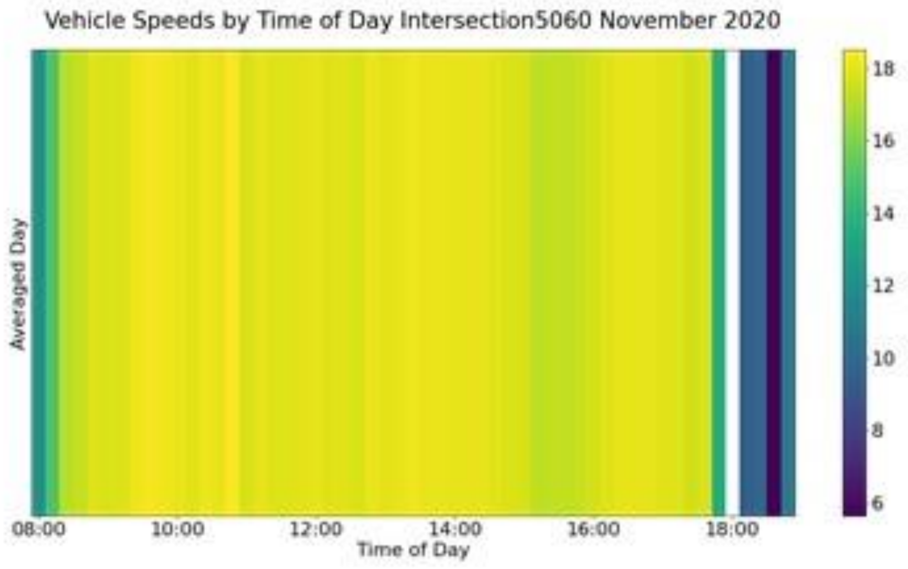


Figure 6-15. Each vertical rectangle represents the average number of vehicles for that time of day in a given month.

- Speeds remained relatively uniform throughout individual days but were generally slower during the weekdays as shown in Figure 6-14 and 6-15.
- Sunday saw the highest average speeds of any day, likely due to decreased congestion and an increase in likelihood of Arrival on Green.

- Friday and Saturday around 6 PM also saw a modest increase in observed speeds, suggesting an increased possibility of fatal collisions during this time.

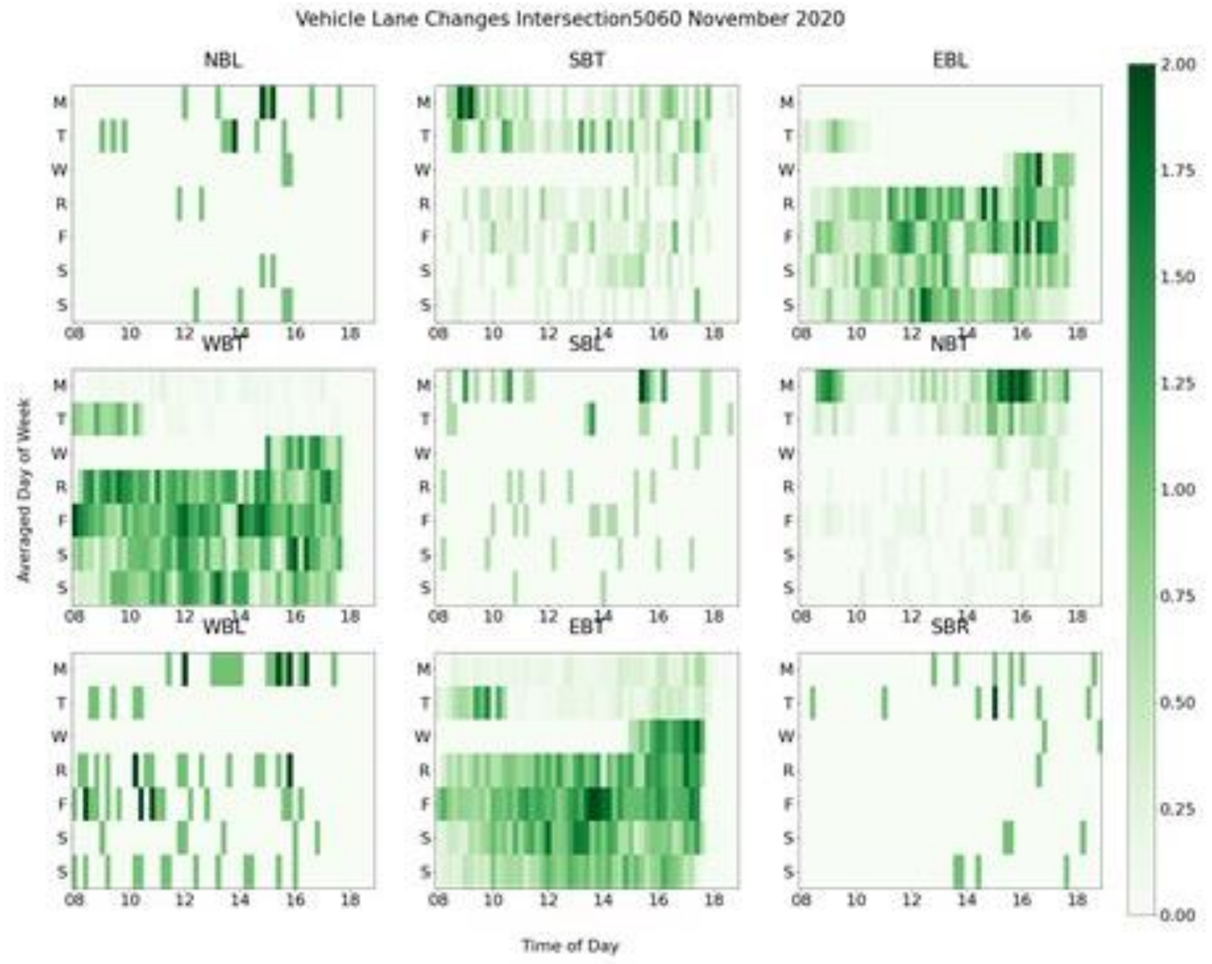


Figure 6-16. Lane changes at an intersection for different time periods. East- and westbound vehicles tend to change lanes most often. One reason is there is a gas station right at the intersection. So many vehicles intending to enter the station slow down traffic,

- Lane changes were most prevalent in through phases, with an abnormal number of them occurring Monday through Wednesday in the eastbound through phase, and to some extent eastbound left, on intersection 5060 (Figure 6-16).

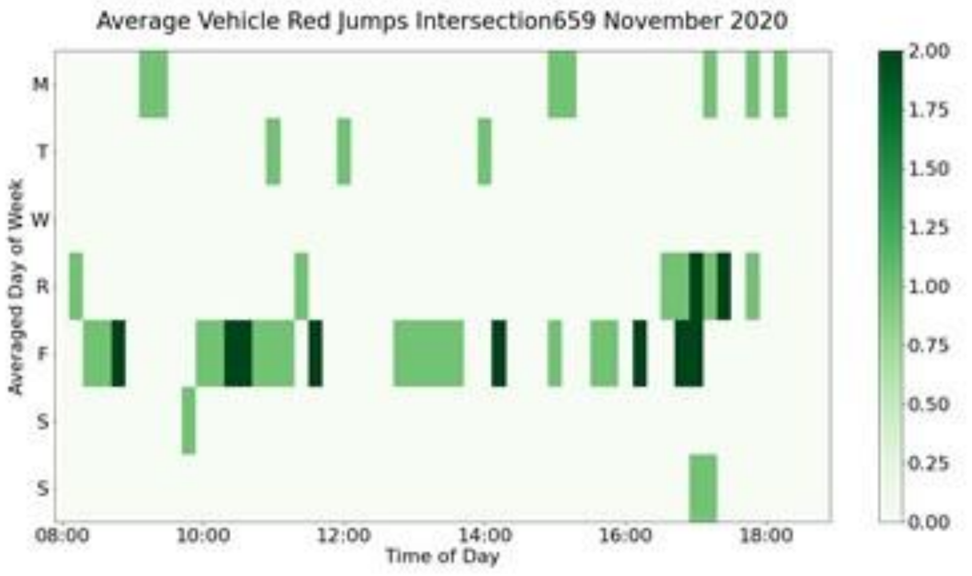


Figure 6-17. A distribution of vehicles that jump the red light by day of week and time of day.

- Red jumps (crossing of intersection on red) are by far most prevalent on Fridays according to our observations. This is shown in Figure 6-17. The factors for this are unclear. These occurrences are relatively rare, so drawing conclusions may be premature until more data are aggregated.

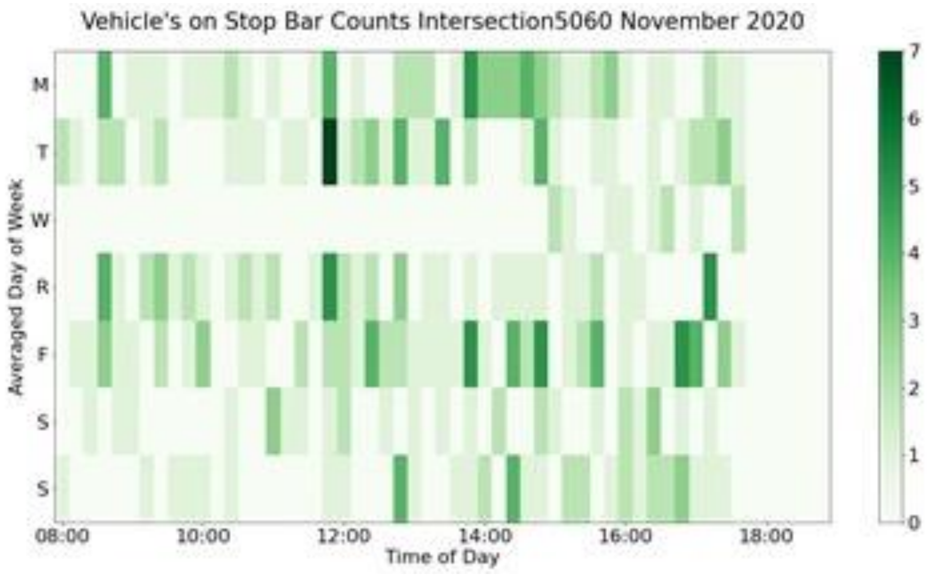


Figure 6-18. Vehicles waiting at the intersection's stop bars. These are mostly vehicles waiting to take a right while yielding to the pedestrian traffic.

- Vehicles waiting on the intersection's stop bars are reasonably distributed throughout the day and week but with an uptick on Fridays, as shown in Figure 6-18. This happens due to high pedestrian volumes.

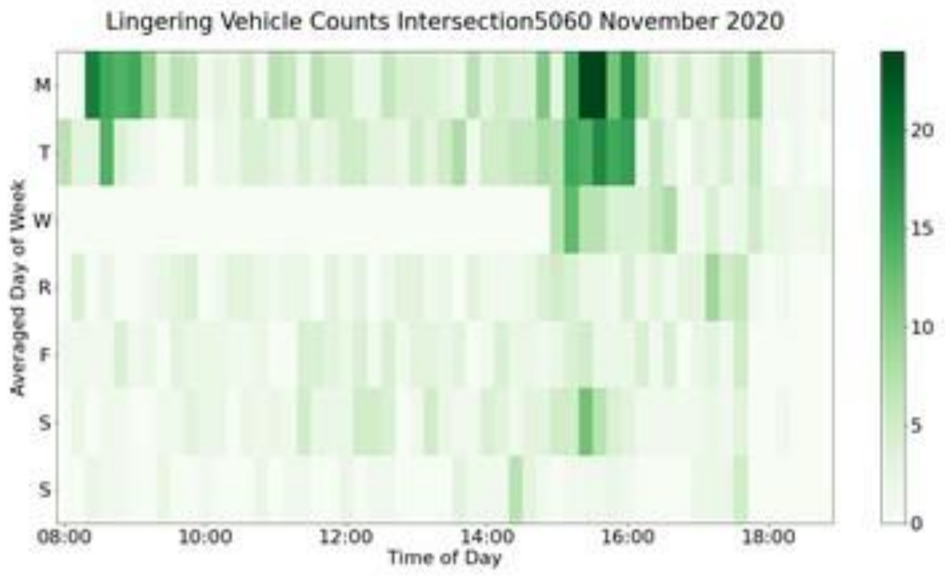


Figure 6-19. Vehicles that drive at very slow speeds within the intersection. These may be left turn vehicles yielding to pedestrians or to through traffic, or right turn vehicles yielding to pedestrians.

- We define lingering vehicles as those who enter the intersection and remain there with low speeds. This may be the case during left turns where vehicles inch out to better cross during the clearing of the through traffic that faces them. This is shown in Figure 6-19. The 8 AM and 4 PM spikes early in the week may correspond with drivers' knowledge that not making the turn during these hours would mean a long wait before getting another turn to cross. This may be an indicator of driver impatience motivated by increased congestion.

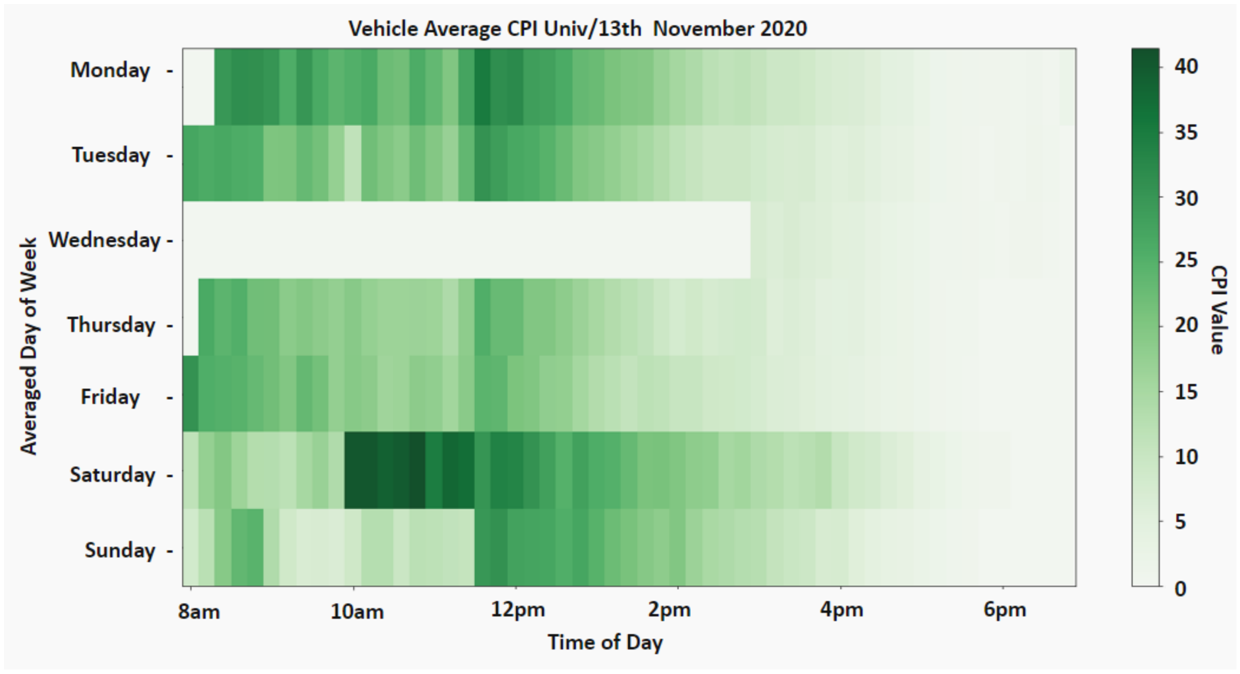


Figure 6-20 Crash Potential Index (CPI) of vehicles at the intersection, over different time periods

- Crash potential index (CPI) peaked on weekends and saw considerable increases during lunch time on weekdays, as can be seen in Figure 6-20.
- Tracking with low vehicle counts and high vehicle speeds, CPI was much higher on weekends than weekdays.
- CPI also depends on the relationship of vehicle speeds and distance between vehicles, which likely explains why the most congested period was also the safest. This may suggest an inverted U-shaped relationship between number of vehicles on the road and risk of collision.
- Anomalous trajectories are reported in Figure 6-21. Upon further analysis it was found that these anomalies happened due to lane changes at the intersection.

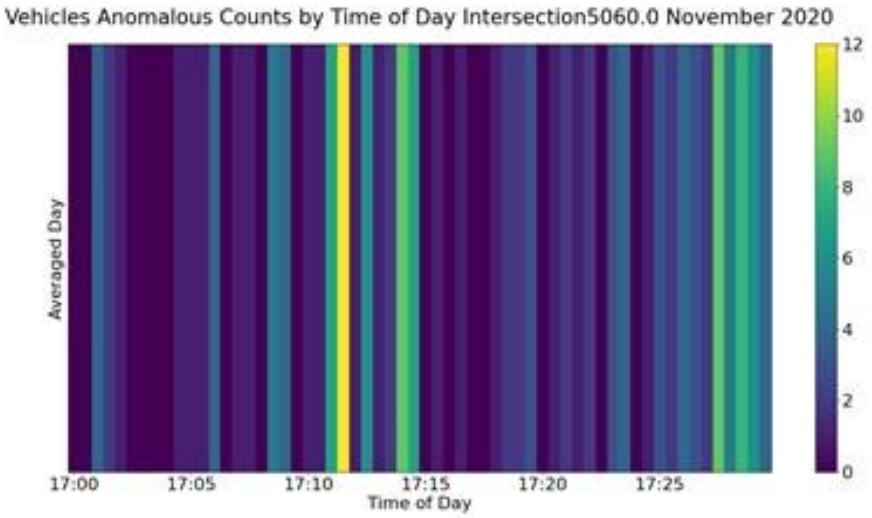


Figure 6-21. Anomalous vehicle trajectories. The anomalies are due to lane changes.

6.4. Conclusion

As part of this task, we present various analysis we performed to analyze the safety characteristics for an intersection. As the video processing pipeline is improved and refined, the insights drawn from our analysis pipeline can be further tested and validated as a tool for evaluating intersection tendencies and safety. The software developed as a part of this task will help us to aggregate more data and compare safety across different intersections to make more powerful generalizations about traffic safety and to begin giving recommendations for improving traffic policy. Weekends present the greatest danger in terms of vehicle-to-vehicle collision, but when combined with our pedestrian analysis, Mondays may be the most dangerous overall. Our final steps will be to observe how modifications in intersection policy affect these metrics using before-after studies and determine if they are sufficient for overall intersection safety.

Chapter 7 – Develop Algorithms and Software for Intersection Incident Detection

7.1. Introduction

Road intersections, where two or more roads meet and cross, show complex geometry and traffic rules, and require drivers' timely maneuvers. It was reported by Federal Highway Administration (FHWA) that more than 50 percent of fatal and injury causing crashes occur at or near intersections. Of all the intersection-related crashes, only about 4 percent is caused by vehicle or environment reasons with 96 percent attributed to drivers. Possible driver-attributed crashes at intersections include inadequate surveillance, false assumptions regarding other driver actions, obstructed views, illegal maneuvers, internal distractions and misjudgments of gaps, speed, etc. Potential solutions for reducing crashes tend to be focused on providing driver assistance of vehicle control or giving timely warnings of potential risks. To this end, predictive algorithms are proposed. For example, by tracking the surrounding vehicles, one can estimate their intended maneuvers, predict their future motion and estimate the risk of collision. Previous studies have been focused on automated vehicle (AV) and advanced driver assistance system (ADAS) solutions. These approaches rely on on-board sensors, such as LIDAR, radar, and vision sensors. However, the on-board sensors have certain limitations, such as limited detection range or field of view, low resolution, drifting position estimation, and sensor imprecision. Therefore, we proposed to incorporate assistance from existing and planned intersection video infrastructure to improve intersection safety.

Vehicle-to-infrastructure (V2I), as the next generation of Intelligent Transportation Systems (ITS), exchanges data from vehicles and infrastructures through wireless communication, enables real-time assistance and warning from infrastructure to vehicles on the road. With the rapid development of wireless communication technologies, especially the recent progress in 5G networks, algorithms running on the infrastructure side are ready to be delivered to road participants and will be highly needed in the foreseeable future.

In this work, we tackle the problem of robust and flexible vehicle trajectory prediction at intersections from surveillance videos, aiming to give early warnings to vehicles about possible collisions and abnormal behaviors. Our proposed approach only requires vision sensors, no signal data is needed, and can be applied to both signaled and unsignalized intersections. Intersections are of different shapes and geometries and are equipped with heterogeneous cameras. A setup stage is needed for each intersection: (1) Google map alignment, (2) learning typical motion patterns from historical data, and (3) training the trajectory prediction model. After the setup, our model can make real-time predictions of all the vehicles' trajectories within the intersection. We also consider the noise introduced by the automatic vehicle tracking algorithms and varying lengths of trajectories captured.

We find the typical motion patterns using a two-step clustering approach. We then find a representative for each cluster as prototype trajectories. Those prototypes act as a strong prior for trajectory prediction algorithms. Each observed partial trajectory can match with the prototypes and be assumed to follow the curve of the prototype with a small deviation, as vehicles would normally follow a certain lane for a certain maneuver. We then introduce a curvilinear coordinate system (CCS), with one axis along the prototype, and the other lateral to the prototype. The CCS largely simplified and reduced the complexities of trajectory prediction models. A long short-term memory encoder-decoder (LSTM-ED) model was designed to predict the future trajectories given its cluster memberships and the observed trajectory represented in CCS.

To summarize, the main contributions of this task are listed as follows:

1. We present a real-time capable vehicle trajectory prediction from surveillance cameras, no traffic signal or GPS data is needed
2. Our algorithm automatically learns the typical motion patterns from historical data, and representatives of them are used in the online phase to speed up and boost the performance of trajectory prediction
3. We design a deep-learning-based trajectory prediction model, which can deal with variable-length observation and prediction period
4. Multiple reasonable predictions are given by the model if multiple motion patterns are likely to describe the observed trajectory

The rest of this report is organized as follows. Section 8.2 gives an overview of our proposed approach, consisting of an offline learning phase and an online prediction phase. Section 8.3 describes the methodology of the offline trajectory clustering and prototype generation. Section 8.4 presents the CCS transformations and the trajectory prediction model. Section 8.5 reports our experiment settings, and quantitative and qualitative results. Finally, Section 8.6 gives a conclusion of our work.

7.2. System Overview

In this section, we will give a procedural description of our algorithmic workflow. It consists of two phases: (1) offline learning and (2) online prediction. Figure 7-1 concisely summarizes the workflow.

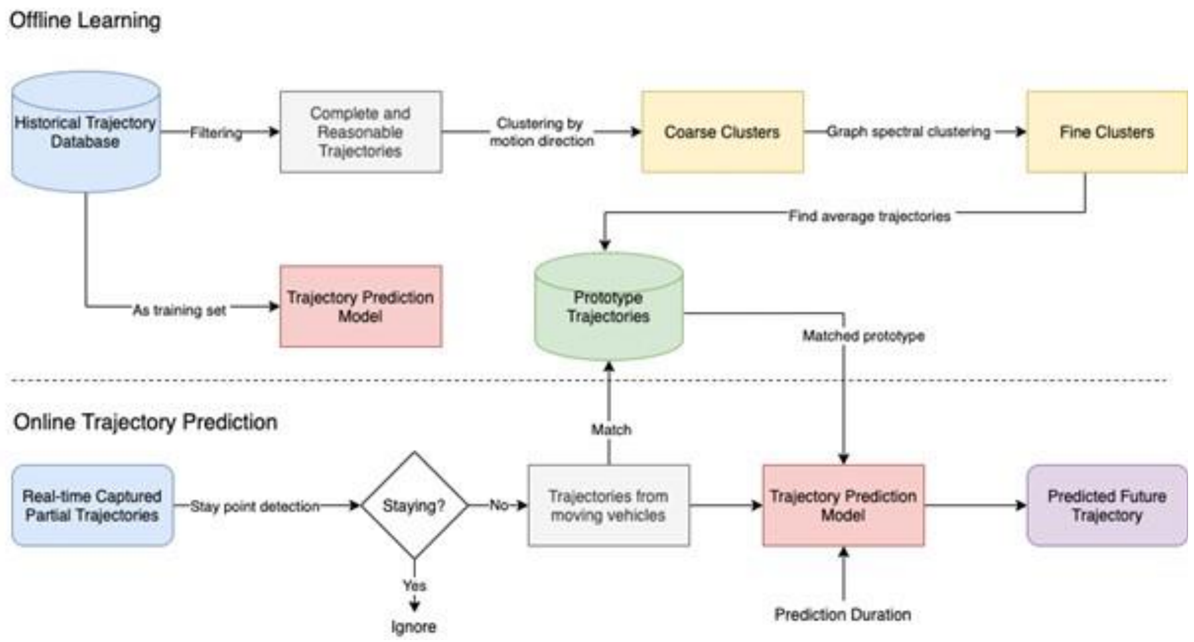

Offline Learning
Historical Trajectory Database
Filtering
Complete and Reasonable Trajectories
Clustering by motion direction
Coarse Clusters
Graph spectral clustering
Fine Clusters
Find average trajectories
As training set
Trajectory Prediction Model
Prototype Trajectories
Matched prototype
Online Trajectory Prediction
Match
Real-time Captured Partial Trajectories
Stay point detection
Staying?
No
Trajectories from moving vehicles
Trajectory Prediction Model
Predicted Future Trajectory
Yes
Ignore
Prediction Duration

Figure 7-1. The overall workflow of trajectory prediction with offline and an online phase. During the offline phase, the trajectories are filtered, clustered and the centroids are determined. Using the centroids, during the online phase the LSTM-based trajectory

7.2.1. Offline Phase

The offline phase can be done periodically to capture the dynamics of traffic evolving (e.g., road construction that causes vehicle detour) or done once for setup if the intersection geometry remains unchanged. There are two main tasks in the offline phase: (1) finding common motion patterns and their representatives and (2) training the LSTM-ED trajectory prediction model.

The trajectories are captured by a vision-based tracking algorithm and are stored in the database. As some tracks in the database are incomplete due to occlusion or physically impossible due to tracking errors, we first filter the dataset to obtain reasonable trajectories. Then we perform a course-to-fine clustering algorithm and average the trajectories in each cluster as the prototypes. The prototypes are stored in the database for online usage. The LSTM-ED model training also happens in the offline phase.

7.2.2. Online Phase

The online phase receives real-time captured trajectories from the online tracking algorithm. We first detect whether the vehicle is waiting for traffic signals using a stay point detection algorithm. We start making predictions when the vehicle starts moving. For a partial trajectory from a moving vehicle, we match with prototype trajectories from the offline phase based on distance measurements. We then feed the likely cluster belonging, matching prototypes, and the partial trajectory to the LSTM-ED to predict its future trajectory.

7.3. Trajectory Prediction Model

In this section, we present a summary of our trajectory prediction model. The trajectory coordinates are first converted from a rectilinear coordinate system (ICS) to a curvilinear coordinate system (CCS). These trajectories are then inputted to the LSTM-ED model for training and inference. We describe the architecture, training and inference using LSTM-ED.

LSTM Encoder-Decoder Model – LSTM networks are designed and proven effective for sequence modeling and prediction tasks. Thus, they are well-suited for trajectories represented as a sequence of coordinates. We adopt the encoder-decoder architecture to cope with variable-length observation and prediction period, as the trajectories captured at an intersection are of variable length. The encoder encodes observed partial trajectories to a fixed-length internal representation, and the decoder decodes the state and predicts possible future motions for a given period. Besides the internal vector from encoder, cluster belonging is also provided as input to the decoder. In this way, the decoder learns to predict differently considering its cluster.

Network Architecture – Both the encoder and decoder consist of two layers: a fully connected (FC) layer and an LSTM layer. The FC layer serves as an embedding function that embeds locations into a fixed-length vector. The embedding will then be fed to the LSTM layer.

Training – We adopt the L2 loss for training, which measures the distance between the predicted and the ground-truth trajectories. As vehicles pass an intersection at different speeds, the number of trajectory points captured in the intersection vary over a wide range. For example, a left turn trajectory usually has more trajectory points captured than a straight heading trajectory, as a left turn vehicle will slow down as it enters the intersection, while a straight heading vehicle tends to be at the maximum speed limit. For this reason, we enable the model to encode variable-length observations and to decode variable-length predictions.

Inference – At inference time, the cluster class of a trajectory *i* is unknown. The model first infers cluster class by matching with all cluster prototypes. The degree of belonging of trajectory to cluster class is calculated by the reciprocal of the distance of the trajectory from the different prototypes.

7.4. Experiments

We evaluated the proposed approaches on the collected trajectory dataset from surveillance fisheye cameras at 3 intersections. We describe below our data collection methods as well as pre-processing steps to obtain Google map aligned trajectories. Following that we compare our trajectory prediction model with baseline methods. Finally, we evaluate the trajectory prediction pipeline for variable-length observation and prediction period.

Data Collection and Preprocessing – Our intersection trajectory dataset is obtained from three busy intersections in Gainesville, Florida. The surveillance cameras have a circular fisheye lens with 10 fps frame rate. From our previous tasks, we have the mapping from fisheye video pixels to Google Maps locations. A detection and tracking pipeline is running periodically to automate the trajectory capture process. We clean the dataset using rule-based filtering. We impose speed limits, within the intersection and non self-intersecting constraints on trajectories. We also applied a trajectory smoothing algorithm to compensate for detection imprecision. In addition, as the traffic signal phase is unknown in our setting, we exclude the stay points of trajectories before entering the intersection. After the above process, we obtain roughly 15,000 trajectories for each intersection which are then split it into training, validation and testing set with a roughly 7:1:2 ratio. The validation set is used in LSTM-ED training to avoid over-fitting. We refer to the three intersections simply as Intersection 1, Intersection 2, and Intersection 3. These are respectively the following intersections in Gainesville: Tower Rd at W University Ave; Museum Rd at Newell Dr; and W University Ave at SW 13th St.

Evaluation of Trajectory Prediction Model – In this section, we evaluate the performance of the proposed trajectory prediction model and compare it with baseline models. This experiment is performed on Intersection 1, and we assume the ground truth cluster class of each trajectory is known. We compare the proposed model (LSTM-ED CCS) with two baseline models:

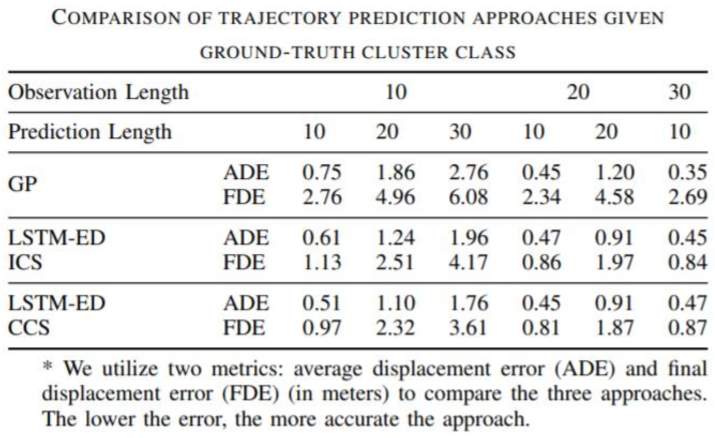
1. LSTM-ED ICS: an LSTM encoder-decoder model without coordinate transformation, otherwise the same as LSTM-ED CCS.
2. GP: a Gaussian process regression model used in Goli et al. (2018).

We adopt the following two metrics for prediction evaluation:

1. Average displacement error (ADE): the average of Euclidean distance between ground truth and prediction over all trajectory points.
2. Final displacement error (FDE): the Euclidean distance between the end positions of ground truth trajectory and predicted trajectory.

We show the prediction errors of different observation and prediction periods. To ensure fair comparison for different periods settings, we set a threshold for minimum trajectory length for evaluation. In our case, only trajectories with more than 40 timestamps are chosen, as the maximum of observation timestamps plus prediction timestamps is 40, as shown in Table 7-1. In this way, the same set of trajectories are used for each setting, as opposed to some trajectories (less than 40 timestamps) only used in shorter period settings. Thus, the experimental result is a true reflection of the model's performance for different observation and prediction periods. The prediction errors are reported in Table 7-1.

Table 7-1. Comparison of trajectory prediction approaches



COMPARISON OF TRAJECTORY PREDICTION APPROACHES GIVEN GROUND-TRUTH CLUSTER CLASS

Observation Length		10			20		30
Prediction Length		10	20	30	10	20	10
GP	ADE	0.75	1.86	2.76	0.45	1.20	0.35
	FDE	2.76	4.96	6.08	2.34	4.58	2.69
LSTM-ED ICS	ADE	0.61	1.24	1.96	0.47	0.91	0.45
	FDE	1.13	2.51	4.17	0.86	1.97	0.84
LSTM-ED CCS	ADE	0.51	1.10	1.76	0.45	0.91	0.47
	FDE	0.97	2.32	3.61	0.81	1.87	0.87

* We utilize two metrics: average displacement error (ADE) and final displacement error (FDE) (in meters) to compare the three approaches. The lower the error, the more accurate the approach.

We find that LSTM-ED CCS outperforms the two baseline models in almost every setting. GP tends to have a large FDE and the prediction result also looks unsmooth. LSTM-ED ICS performs similarly on shorter prediction periods as LSTM-ED CCS, but worse on longer prediction periods.

Evaluation of Trajectory Prediction Pipeline – Our pipeline consists of cluster class determination and trajectory prediction. In other words, unlike the previous experiment, the ground truth cluster class is unknown to the LSTM-ED model. We evaluated the pipeline on all three intersections. The quantitative result is reported in Table 7-2. When 20 or 30 observations are given, the model usually gives ADEs of less than 1 meter, and FDEs of less than 3 meters. In the most extreme case of our settings, when only 10 observations are available, and 30 predictions are required, the ADE is about 2 meters and FDE is about 4–5 meters. The FDE here is relatively high because 10 observations is usually not enough to decide a trajectory's cluster (for example, some lanes can either go straight or turn right, it's hard to tell a vehicle's intention from only 1 second of data). Overall, our pipeline performs reasonably well on all three intersections.

The qualitative result is shown in Figure 7-2. Our model produces multiple outputs if the observed trajectory matches with multiple prototypes. Figure 7-3 shows one example of multiple outputs. From the observed trajectory, going straight and taking a right are both likely.

Table 7-2. Prediction errors for Intersections 1, 2, and 3

Observation Length		10			20		30
Prediction Length		10	20	30	10	20	10
Intersection 1	ADE	0.54	1.21	2.09	0.52	1.16	0.61
	FDE	1.02	2.70	4.83	0.97	2.61	1.15
Intersection 2	ADE	0.54	1.23	2.05	0.46	0.91	0.49
	FDE	1.03	2.72	4.29	0.79	1.90	0.82
Intersection 3	ADE	0.66	1.40	2.19	0.58	1.18	0.54
	FDE	1.24	2.89	4.38	1.03	2.44	0.99

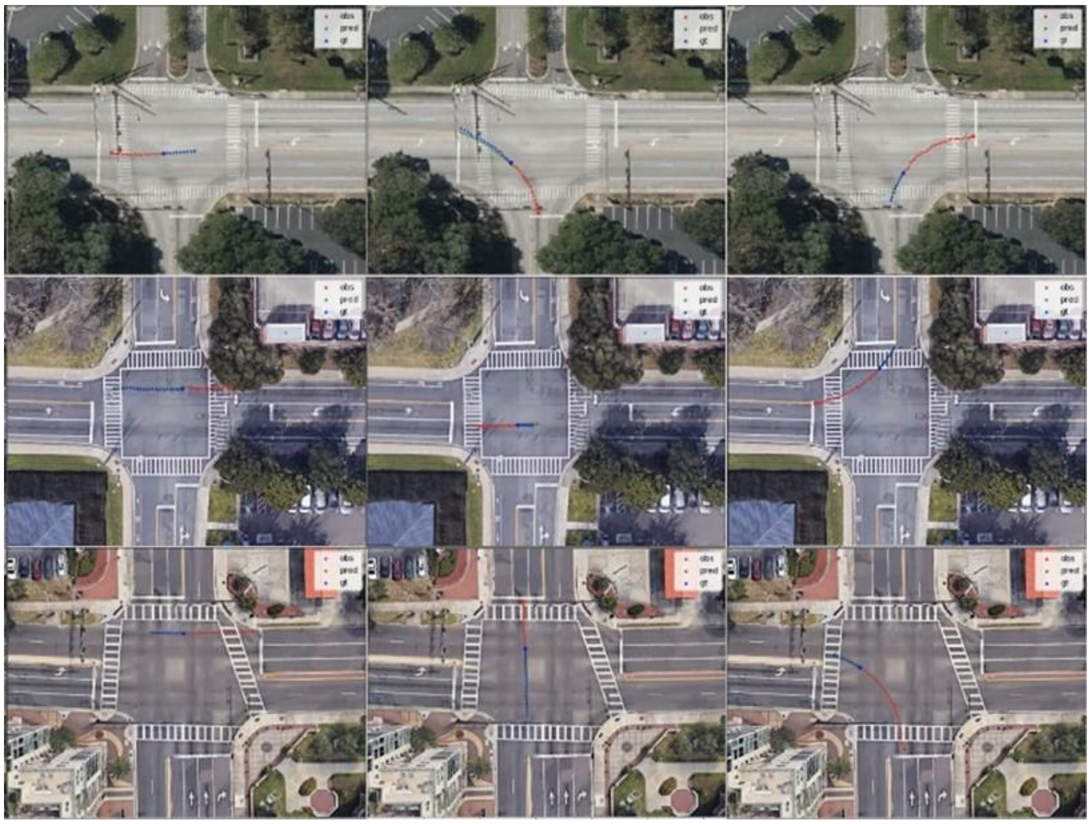


Figure 7-2. Samples of the prediction results.

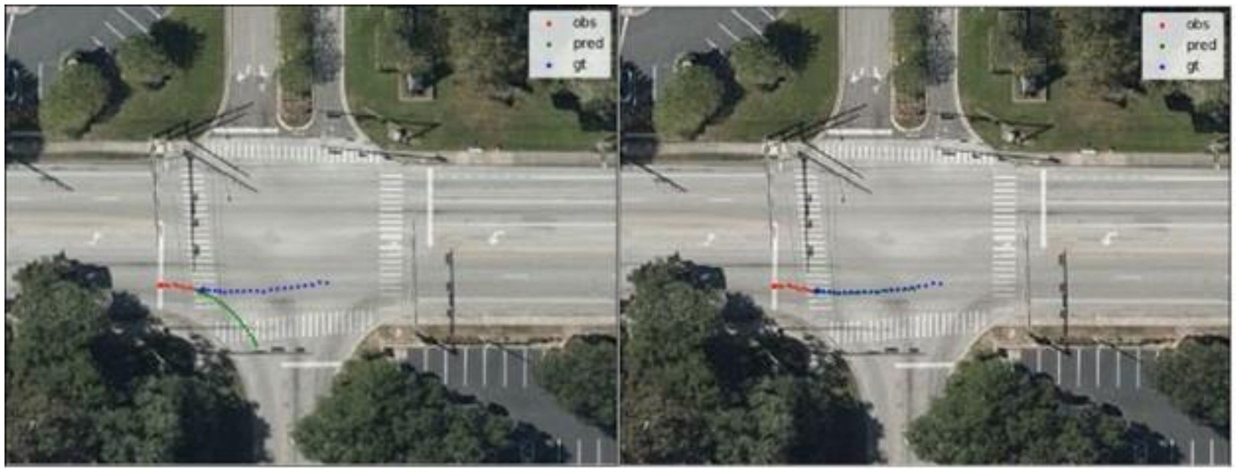


Figure 7-3. Two predictions at inference time. The red, blue and green points represent observed, ground-truth future, and predicted points of a trajectory, respectively.

7.5. Conclusion

A real-time trajectory prediction approach coupled with aligned google map information is proposed in this paper. Our approach makes use of a historical trajectory database and finds typical motion patterns as a guide for future prediction. Given this prior information, our approach can make reasonable predictions based on variable starting position, observation period and prediction period. Experimental results on three intersections show the effectiveness and extensibility of our approach. In the immediate future, we plan to integrate our trajectory prediction module to an early warning system. We believe our work will help increase the intersection safety.

Chapter 8 – Implement and Test Edge-based Software for Near-misses, Anomalous Behavior, and Incidents

8.1. Introduction

It was reported by the Federal Highway Administration (FHWA) that more than 50% of fatal and injury-causing crashes happen at or near intersections, among which 96% come from drivers' inadequate surveillance (such as internal distractions, blind spots/occlusions), or erroneous judgment of other road participants' attentions. Advanced driver assistance system (ADAS) solutions have been proposed and improved over the years; however, onboard sensors still suffer from frequent view occlusion, sensor imprecision, failure in new environments, etc. Fortunately, these shortcomings can be filled in with more complete and robust data generated from overhead video sensors that possess better vantage points for road participant analysis. Sensors can then capture and relay any missing and potentially consequential information to road participants so they can react appropriately.

This chapter is organized as follows based on the two broad subtasks. The first subtask is developing a systematic and comprehensive method of analyzing intersection safety detailed in Section 9.2. The second subtask is deployment of our predictive near-miss algorithm on edge devices which includes measurement of the latency of video processing and generating alerts for potential near-miss events. We report on the second subtask in Section 9.3, and finally, we conclude the report in Section 9.4.

8.2. Safety Analysis of Intersections

We outline our methodology here by first describing the current strategy for intersection safety analysis and introducing the classification scheme we use for the conflicts. Then, we describe our video processing technique, the computation of safety related features, and the subsequent filtering and classification of events using these features.

Through this work, we have made several contributions to the field of intersection safety analysis, as follows:

1. The combination of video and signal timing data, giving a more comprehensive view of conflict event occurrence in terms of both temporal and spatial specificity.
2. The ability to determine trajectory movement enables which enables accurate event classification in terms by conflict type.
3. Separate safety analysis for pedestrian-to-vehicle interactions and vehicle-to-vehicle interactions.
4. Introduction of a new surrogate safety measure, "severe event," which is quantified by multiple existing metrics such as time-to-collision (TTC), post-encroachment time (PET), deceleration, and speed.
5. An efficient multistage event filtering approach to determine severe events.

This section is organized as follows. Section 9.2.1 presents the methodology used to generate trajectories and compute features from the trajectories. Section 9.2.2 presents the existing strategy of classification of serious conflicts in the literature and introduces a new surrogate safety measure, "severe events," that we developed as a part of this work. Section 9.2.2 also presents sets of filters used to both discard non-events and then detect severe ones. Section 9.2.3 presents our experimental results.

8.2.1. Generating Features from Trajectories

The video processing pipeline takes fisheye video footage as input, annotates objects with bounding boxes, maps the coordinates from the fisheye to rectilinear space, and then stores the results in the trajectory table. The object detection and tracking module utilizes You Only Look Once version 4 (YOLOv4) to detect different kinds of road participants, including vehicles, pedestrians, cyclists, and motorcyclists. Additionally, a modified Deep Sort algorithm is used to associate detections across frames and assign a unique ID for each object. Due to the fisheye distortion of recorded trajectories, we perform rectification and alignment to Google Maps images before feeding the trajectories to downstream modules. Our solution for rectification includes two steps: fisheye-to-perspective transformation followed by thin-plate splines (TPS) warping.

Separately, (ATSPM) data provided by the city is collected, which provides the signal information for each traffic light per intersection over time. These signals are merged with object trajectories, which enables analyses that are concerned with object location and signal states over time. such as signal violations, and lingering mid-trajectory. One issue that arises however is the desynchronization of city provided signal changes and video-recorded signal changes which usually varies by a few seconds. A purpose-built computer vision model is trained to mark traffic signal states and is compared to ATSPM signal changes to yield the time delay between video and city data. If a signal is not visible, then the start-up time of a vehicle is used under the assumption that driver reaction time is roughly one and a half seconds.

We now have a comprehensive set of features computed from every potentially severe event. In this section, we describe the feature set. The following is a list of key features that we compute:

1. Standard Near-Miss Attributes: We compute the common risk assessment metrics such as TTC and the PET for every event. We also compute a derived metric known as the crash potential index (CPI). TTC is "the time required for two vehicles to collide if they continue at their present velocity and on the same path" (Hayward, 1972), PET is the difference in time between when the first road user leaves the conflict point and the second road user arrives at the conflict point (Allen et al., 1977), and CPI is "the probability that the deceleration rate to avoid a collision (DRAC) exceeds MADR at a moment" (Mahmud et al., 2017), where MADR is the maximum available deceleration rate.
2. Phase Information: The fused video and ATSPM dataset allow us to include the signal-related features such as the ongoing vehicle phase, ongoing pedestrian phase, and when the event occurs relative to the ongoing phase (flagged as during the beginning, middle, or end); during the beginning, middle or end of the current phase.
3. Trajectory Features: Then we include the trajectory-related features such as the movement of trajectories that allows us to tell whether the conflict was a crossing, merging, or diverging conflict, or whether it was a same-lane leading-following conflict. In the case of pedestrian-to-vehicle conflicts, trajectory features enable us to tell if the conflict was due to a left-turning vehicle or a through vehicle or a right-turning vehicle in the adjacent parallel crosswalk or near-side perpendicular crosswalk.
4. Speed Features: We compute speeds and accelerations over time for both vehicles in a vehicle-to-vehicle conflict and we record the timestamp of the maximum deceleration values if there is a deceleration in one or both vehicles.

5. Distance Features: Distance-related features between two vehicles or between the vehicle and the pedestrian at the time of the interaction are other metrics that we compute and store for later use.
6. Meta-information: We store the road user class such as car, bus, truck, motorcycle, or pedestrian. We also collect and store other information such as unique ID assigned to the road user by video processing, the location of each user at the time of the incident, and intersection properties such as size, median width, location (city and state), the total number of road users for one minute around the time the event occurs. We also store the timestamp and secondary features derived from the timestamp, such as day of week and hour of day.

8.2.2. Classification of Severe Events

The safety of roadways is measured using crash rates and crash severity. Crashes involve a vehicle colliding with another vehicle, a pedestrian, or some other object on or by the roadside. The severity of crashes is measured on a five-level scale based on the most serious injury suffered by anyone involved in the crash: K (fatality), A (incapacitating injury), B (non-incapacitating injury), C (minor injury), and O (property damage only and no injuries). While we acknowledge that one crash is still one too many, their relative rarity makes drawing valid safety-related conclusions difficult and necessitates using possibly outdated data. Alternatively, predictive models (which were themselves developed based on historical crash data from several years) can be applied to determine the expected number of crashes on roadway facilities.

Based on decades of safety research using crash data, it is generally acknowledged that using "surrogate measures of safety" could provide further insights into enhancing safety of roadways. These surrogate measures rely on maneuvers (trajectories) of vehicles and pedestrians. By understanding the nature of trajectories that could have led to a crash, countermeasures can be developed to reduce or even eliminate such unsafe maneuvers. The most common surrogate safety measure is the "near-miss" or the "traffic conflict." Near-misses involve a vehicle's trajectory coming "very close" to that of another vehicle or a pedestrian without an actual collision happening. The proximity of the trajectories is measured on a temporal scale using metrics such as time-to-collision (TTC) and post-encroachment time (PET). The severity of the near-miss event can be determined based on both the temporal proximity measures (TTC and PET) and the velocities of the vehicles or pedestrians involved. Typically, shorter TTC and PET combined with higher relative velocities imply a diminished ability to avoid collision and a higher potential for injury. Unlike the case of crashes, there are currently no clear thresholds or categories for classifying near-misses by severity. In this report, we introduce a new surrogate safety measure, "severe event," that includes all near-miss events as well as unsafe behavior exhibited by road users.

While surrogate safety measures offer a great opportunity to understand site-specific and time-specific safety issues and develop countermeasures, a great practical impediment in using surrogate measures for safety analysis is the need to process large volumes of video data to determine trajectories, identify the conflicts in these trajectories, and filter these down to a subset of critical unsafe maneuvers for further analysis. In the context of signalized intersections, it is also necessary to analyze the unsafe maneuvers with respect to the ongoing signal phasing data to identify the appropriate countermeasures. For example, unsafe maneuvers frequently happening during a permitted left-turn phase may suggest the need for a protected left-turn phase. Similarly, frequent conflicts between right-turning vehicles and crossing pedestrians may suggest separating the signal phases for these two movements (no right turn on red or leading pedestrian phase).

The substantive focus of this report is on processing hours of video data to identify and characterize unsafe maneuvers involving isolated users as well as multiple users at signalized intersections.

- Single trajectory: Unsafe or illegal trajectories involving only a single vehicle or a pedestrian can be useful for identifying potential safety issues at the intersection. For example, a vehicle running a red light or a pedestrian jaywalking when there is no cross-traffic can be considered unsafe maneuvers. While the driver or pedestrian was not at risk during that instance there is no guarantee that repeating such behaviors will still be safe.
- Multiple trajectories: Unsafe maneuvers involving multiple users at signalized intersections are broadly classified into two major categories: vehicle-vehicle conflicts and vehicle-pedestrian conflicts, as described below.

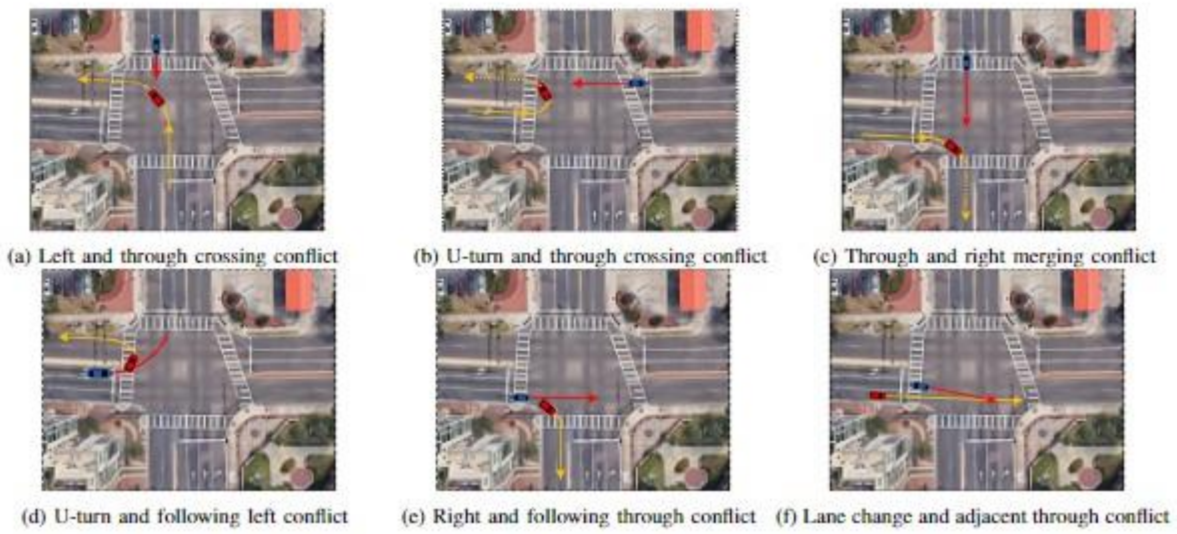

(a) Left and through crossing conflict
(b) U-turn and through crossing conflict
(c) Through and right merging conflict
(d) U-turn and following left conflict
(e) Right and following through conflict
(f) Lane change and adjacent through conflict

Figure 8-1. Feasible vehicle-vehicle conflicts at signalized intersections

1. Vehicle-vehicle conflicts: As the primary purpose of signalization is to reduce or eliminate conflicting movements through the intersection, the following are the feasible conflicts between vehicles at signalized intersections if all vehicles strictly follow the traffic rules (assuming a three- or four-leg intersection, which are the most common):
    - Left turn and opposing through (a left-turning vehicle in a permitted phase conflicts with an opposing through movement such as in Fig. 8-1a)
    - U-turn and opposing through (a U-turning vehicle in a permitted phase conflicts with an opposing through movement such as in Fig. 8-1b)
    - Through and right turn (a right-turning vehicle merging on the same lane as a through vehicle such as in Fig. 8-1c)
    - U-turn and a following left turn (a leading U-turn with a following left-turning vehicle such as in Fig. 8-1d)

- Right turn and a following through (a leading right-turning vehicle with a following through vehicle such as in Fig. 8-1e)
- Lane change and adjacent through (a lane-changing vehicle conflicting with adjacent through such as in Fig. 8-1f)
- Rear-end conflicts when the leading vehicle moves slower than the following vehicle on the same lane.

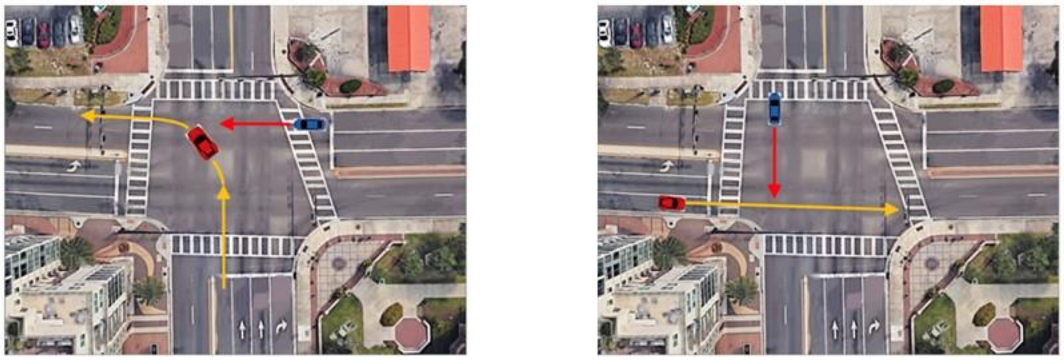


Figure 8-2. Possible illegal vehicle-to-vehicle conflicts: (a) Illegal through and left conflict; (b) Illegal opposing through conflict

If one or more vehicles do not strictly follow the traffic rules (say run the red light), other conflicts are also possible:

- Illegal through and left conflict (see Fig. 8-2a).
- Illegal opposing through conflict (see Fig. 8-2b).

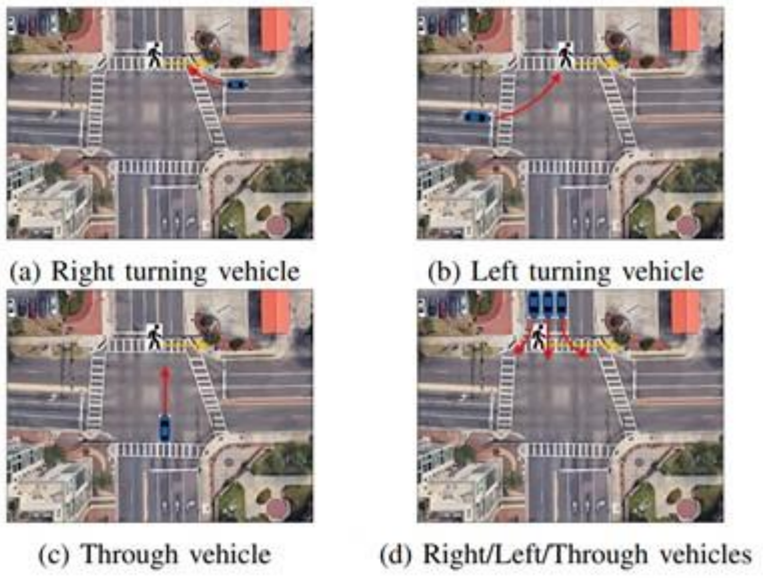


Figure 8-3. Pedestrian-to-vehicle conflicts at signalized intersections

2. Vehicle-pedestrian conflicts: The following are the feasible conflicts between vehicles and pedestrians at signalized intersections if all vehicles and pedestrians strictly follow the traffic rules (assuming a three- or four-leg intersection, which are the most common):
   - Right-turning vehicle with the pedestrian in an adjacent parallel crosswalk (Fig. 8-3a)
   - Left-turning vehicle with pedestrian in the far-side parallel crosswalk (Fig. 8-3b).

   If one or more vehicles or pedestrians do not strictly follow the traffic rules (e.g., run the red light), other conflicts are also possible:
   - Through vehicle with far-side perpendicular crosswalk (Fig. 8-3c)
   - Through or right turning vehicle with near-side perpendicular crosswalk (Fig. 8-3d)

While the vehicle-vehicle and vehicle-pedestrian conflicts so described are mainly between a pair of road users, they could easily lead to multiple pairwise events in quick succession, resulting in a cluster of road users being affected.

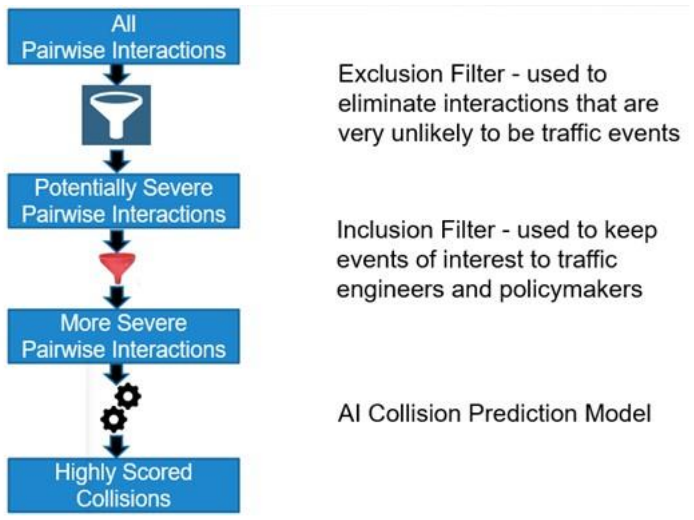

All Pairwise Interactions
Exclusion Filter - used to eliminate interactions that are very unlikely to be traffic events
Potentially Severe Pairwise Interactions
Inclusion Filter - used to keep events of interest to traffic engineers and policymakers
More Severe Pairwise Interactions
AI Collision Prediction Model
Highly Scored Collisions

Figure 8-4. Event-filtering sieve

Figure 8-4 shows the event-filtering sieve developed to filter the traffic interactions to keep only the severe events. We describe each stage of the filtering process below.

1. Exclusion filter: Due to the quadratic nature of object pair comparisons, we first apply filters to exclude pairs with near-zero chance of being an event of interest. Then we then compute safety metrics and filter once more to isolate and store sufficiently severe interactions for further analysis. These filters are called exclusion and inclusion filters, respectively.

   For vehicle-only interactions, exclusion filters check whether the vehicles are conflicting in phase or lane and, in the conflicting phase case, whether they have a conflict point within the intersection, whether the computed TTC or PET is less than 10 seconds, whether the vehicles are moving faster than a small threshold (1 mph in our case), or for a same lane leading following TTC conflict, whether the relative speed between the vehicles is greater than 10 mph. Also, for TTC computations, whether the vehicles are inside the intersection polygon together, are within a threshold distance of each other (10 meters in our case).

   For pedestrian-to-vehicle interactions, the exclusion filter keeps for further analysis, all interactions where pedestrian and vehicle phases are conflicting, and when the distance between the pedestrian and the vehicle is 10 meters.

2. Inclusion filter: After exclusion filtering is complete, the resulting pool of interactions likely still include a large percentage of non-dangerous events. Once inclusion filtering is performed, however, the final pool of candidates should constitute mostly severe traffic events, assuming any occurred. For our case studies, we required serious traffic events to be events with a TTC under 2 seconds or PET under 3 seconds and a deceleration greater than 5 feet per second in either case. Lastly, the resulting events can be validated manually, and further threshold tuning can take place if needed.

   For pedestrian-to-vehicle interactions, a proximity threshold between the pedestrian and the vehicle is the primary condition of inclusion, but there are additional conditions depending on the pedestrian's state. If the pedestrian is crossing, then the pedestrian and vehicle trajectories must be conflicting (condition 1). Since jaywalking is inherently risky behavior, for a pedestrian jaywalking, proximity is the only requirement for the event to be considered as potentially severe (condition 2). The threshold values for each of these conditions can also differ to account for the risk inherent in each one of these states. The thresholds for the two conditions used in the case study below were set as 5 and, 3 meters, respectively. If the pedestrian is not crossing, then they must also be near a crosswalk for such events to be considered as potentially severe (condition 3) and in that case we consider the distance between the pedestrian and the vehicle to be 1 meter or less.

3. Human-annotated classifier: Although the usage of primary safety measures was efficient in finding near-miss events, manually reviewing videos still constitutes a large use of human resources. For this reason, we utilize a simple decision tree to predict human-annotated event severity as another layer of filtering. During the filtering process, we use the primary safety measures of TTC, PET, and deceleration for vehicle-to-vehicle interactions and, additionally, phase compatibility and distance for pedestrians. Human ratings of events are used to validate the model's predictions.

8.2.3. Experimental Results

Table 8-1 gives the details of the intersections and the hours of video stream we processed.

Table 8-1. Summary of the intersections being analyzed

Intersection	City	Speed Limit (mph)	Pedestrian Presence (% of total traffic)	Total traffic	Protected / Permissive Left	Dates processed	Total hours processed
WUAve/13thSt	Gainesville	25	18.5	146,133	protected	Oct 16-Dec 17	718.4
WUAve/17thSt	Gainesville	25	41.8	67,550	permissive	Nov 13-Nov 19	102.8
WUAve/20thDr	Gainesville	25	2.9	105,590	permissive	Nov 1 – Nov 7	107.7
23rdAve/55thSt	Gainesville	45	1.6	97,173	permissive	Nov 1-Nov 7	82.9

Table 8-2 gives the number of events processed while applying the exclusion and inclusion filters. The numbers in the brackets represent the pedestrian counts. The last column gives the exposure metric by counting the total number of road users and also the total count of pedestrians in parenthesis.

Table 8-3 gives the count of events after applying inclusion filter per 10,000 vehicles for the study period, while Table 8-4 gives the event counts per 10,000 pedestrians for the study period. We find that for vehicles, the intersection of 23rd Ave at 55th St is the riskiest, while the other intersections are reasonably safe. For pedestrians, W University Ave at 20th Dr (also known as Gale Lemerand Dr) is the riskiest, followed by the intersection of 23rd Ave at 55th St and W University Ave and SW 13th St.

Table 8-2. Events processed while applying the exclusion and inclusion filters

Intersection	After exclusion filter	After inclusion filter	Severe events (manual inspection)	Dates	Total Road Users (Pedestrians)
NW 23rd Ave at 55th St	3178 (71)	1427 (64)	44 out of first 100	11/01–11/07	97,173 (1566)
W University Ave at 13th St	2965 (819)	920 (300)	74 out of first 100	11/09–11/15	146,133 (26,258)
W University Ave at 17th St	441 (209)	287 (152)	8 out of first 29	11/13–11/19	67,550 (28,264)
W University Ave at 20th St	2541 (569)	683 (331)	——	11/01–11/07	105,590 (3071)

Table 8-3. Event counts per 10,000 vehicles for the study period

Intersection	Normalized event counts after applying inclusion filter for vehicle-vehicle interactions	Dates
NW 23rd Ave at 55th St	143	11/01–11/07
W University Ave at 13th St	52	11/09–11/15
W University Ave at 17th St	34	11/13–11/19
W University Ave at 20th St	34	11/01–11/07

Table 8-4. Event counts per 10,000 pedestrians for the study period

Intersection	Normalized event counts after applying inclusion filter for pedestrian-vehicle interactions	Dates
NW 23rd Ave at 55th St	409	11/01–11/07
W University Ave at 13th St	114	11/09–11/15
W University Ave at 17th St	54	11/13–11/19
W University Ave at 20th St	1,078	11/01–11/07

8.3. Real-time Near-miss Detection

Our real-time near-miss detection method detects the safety breaches in real time so that the drivers or pedestrians involved may be alerted. The system relies on real-time wireless communication technology and real-time data processing pipeline. For real-time communication, Vehicle-to-infrastructure (V2I) technologies have been proposed and evolved rapidly in recent years, especially after the invention of 5G networks. V2I technologies exchange data from vehicles and infrastructures through wireless communication and enable real-time assistance and warning from infrastructure to vehicles on the road. For the real-time data processing pipeline, we verified the feasibility, and we introduce our developed pipeline in detail in the rest of this report. Specifically, our pipeline consists of the following modules: object detection, object tracking, trajectory prediction, and near-miss prediction. The input for the pipeline is surveillance video, and the output is real-time SMS messages of potential near-misses. We utilize RabbitMQ for fast data exchange between modules. Our fully developed system latency is only around 1 second.

We have already presented our near-miss prediction algorithm. In Section 9.3.1 of this report, we present the system used to deploy the technology. Then we present our trajectory prediction algorithm briefly in Section 9.3.2. Finally, we present our experimental results and latencies in Section 9.3.3.

8.3.1. System

The system takes fisheye videos as input and runs through the following modules to produce near-miss predictions: (1) object detection, (2) object tracking, (3) trajectory prediction, and (4) near-miss prediction. We adopt a You Only Look Once v4 (YOLOv4) object detector to detect different kinds of road participants, including vehicles, pedestrians, cyclists, and motorcyclists. A modified deep sort algorithm is used to associate detections across frames and assign a unique ID for each object. As the trajectories from fisheye videos are usually of unnatural shapes caused by distortion, we perform rectification and alignment to Google Maps images before feeding the trajectories to downstream modules. Our solution for rectification includes two steps: fisheye-to-perspective transformation followed by thin-plate splines (TPS) warping. Then the trajectories are processed by our trained trajectory prediction algorithm to predict their future movements. The trajectory prediction algorithm makes use of historical trajectories in the database to learn the typical motion patterns in order to make more informed predictions. Finally, a heuristic-based near-miss prediction algorithm is utilized to predict near-miss events. The output of our pipeline is then sent to receivers through SMS messaging.

We adopt RabbitMQ and Apache NiFi for message passing and data monitoring. Figure 8-5 shows the NiFi control panel for the real time near-miss detection pipeline. The detection and tracking algorithms run on two GPUs that correspond to Tracking Core 1 and Tracking Core 2 module, as represented by the top two boxes in the left of Figure 8-5. The tracks are then aligned to the Google Maps image with a top-down view. Then aligned tracks are fed to the track prediction module for near-miss generation. We also utilize a Logs module to store runtime logging and system information. Lastly, the UI front end module reads from the database and visualizes each functionality of our system.

In the rest of this section, we will put our focus on the trajectory prediction algorithm as well as the near-miss prediction module, as they are the focus of this report.

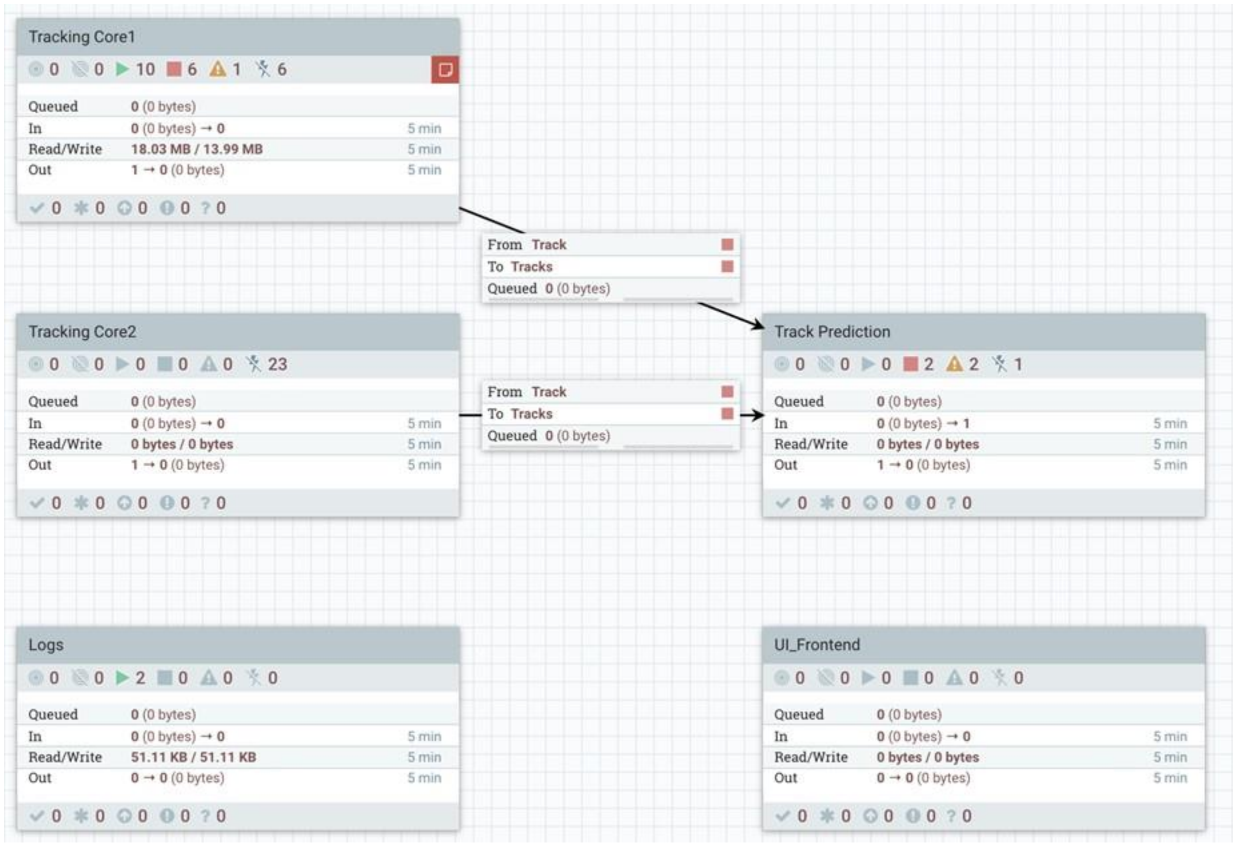
8.3.2. Trajectory Prediction Algorithm

Our trajectory prediction module has an offline learning phase and an online inferencing phase. The offline phase can be done periodically to capture the dynamics of traffic evolution (e.g., road construction that causes vehicle detour) or done once for setup if the intersection geometry remains unchanged. The online phase receives real-time captured trajectories from the online tracking algorithm and makes reasonable predictions for a required time interval. This algorithm has two phases:

1. Offline phase: We use a portion of historical trajectories to learn common motion patterns by a course-to-fine clustering algorithm, followed by finding prototypes to represent each motion cluster. In order to generate longer and smoother prototype trajectories, filtering and smoothing

algorithms are applied before clustering. Given historical data and their corresponding prototype trajectories, a model is trained to complete a trajectory as it develops. We utilize a modified long short-term memory (LSTM) network as the trainable trajectory prediction model, considering its effectiveness of modeling sequential data, robustness to input noise, and flexibility to process variable-length input trajectories.

2. Online phase: We first detect whether the vehicle is waiting for traffic signals using a stay-point detection algorithm. We begin making predictions when the vehicle starts moving. For a partial trajectory from a moving vehicle, we match with prototype trajectories obtained from the offline phase based on distance measurements. We then feed the partial trajectory into the algorithm, matching prototypes into the trajectory prediction model to output its future trajectory.



```
Tracking Core1
0   0   10   6   1   6
Queued       0 (0 bytes)
In           0 (0 bytes) → 0           5 min
Read/Write   18.03 MB / 13.99 MB       5 min
Out          1 → 0 (0 bytes)           5 min
0   0   0   0   0

From  Track
To  Tracks
Queued  0 (0 bytes)

Tracking Core2
0   0   0   0   0   23
Queued       0 (0 bytes)
In           0 (0 bytes) → 0           5 min
Read/Write   0 bytes / 0 bytes         5 min
Out          1 → 0 (0 bytes)           5 min
0   0   0   0   0

From  Track
To  Tracks
Queued  0 (0 bytes)

Track Prediction
0   0   0   2   2   1
Queued       0 (0 bytes)
In           0 (0 bytes) → 1           5 min
Read/Write   0 bytes / 0 bytes         5 min
Out          1 → 0 (0 bytes)           5 min
0   0   0   0   0

Logs
0   0   2   0   0   0
Queued       0 (0 bytes)
In           0 (0 bytes) → 0           5 min
Read/Write   51.11 KB / 51.11 KB       5 min
Out          0 → 0 (0 bytes)           5 min
0   0   0   0   0

UI_Frontend
0   0   0   0   0   0
Queued       0 (0 bytes)
In           0 (0 bytes) → 0           5 min
Read/Write   0 bytes / 0 bytes         5 min
Out          0 → 0 (0 bytes)           5 min
0   0   0   0   0
```

Figure 8-5. NiFi configuration of the real-time near-miss detection pipeline

The near-miss prediction module takes the last 3 seconds of trajectory data and predicts the near-misses in the next 2 seconds. For all trajectories within the 3-second period, the module first applies the trajectory prediction algorithm to find their future trajectories for at most 2 seconds. Next, each trajectory is aligned by timestamp and then pairwise distances are computed. The distance between two trajectories is defined as the minimum distance of the two at any predicted timestamp. The distance being too small suggests a possible near-miss event; thus, we set a threshold (3 meters) and only consider pairs with distance below the threshold as potential near-misses.

However, because the upstream modules are not perfect, there is a noticeable number of false positives if we only consider the trajectory distance criteria. Therefore, we developed several heuristic rules to filter the results and reduce the false positive rate, as shown in Figure 8-6. For pairs of trajectories whose predicted distance is less than 3 meters, we determine their conflict type. We only consider two scenarios: same-lane conflict and conflicting-lane conflict. For the same-lane scenario, a collision happens when the rear car moves much faster than the front car, which results in a rear-end collision. For the intersecting-lane scenario, a collision happens when two vehicles are moving at a relatively high speed. For pairs that pass either the same-lane or intersecting-lane conflict criteria, a time-to-collision (TTC) and a brake deceleration filter are applied. If the TTC between them is less than 2 seconds or one of them has a deceleration more than 5 ft/s², the pair will be determined as a near-miss and a warning is outputted.

We also extend our system to support vehicle-pedestrian near-miss prediction. As pedestrians usually move much slower than vehicles, we omit their motion and only predict vehicles' motions. If the distance between the pedestrian and the predicted vehicle trajectory is less than a threshold (2 meters), we consider it as a near-miss.

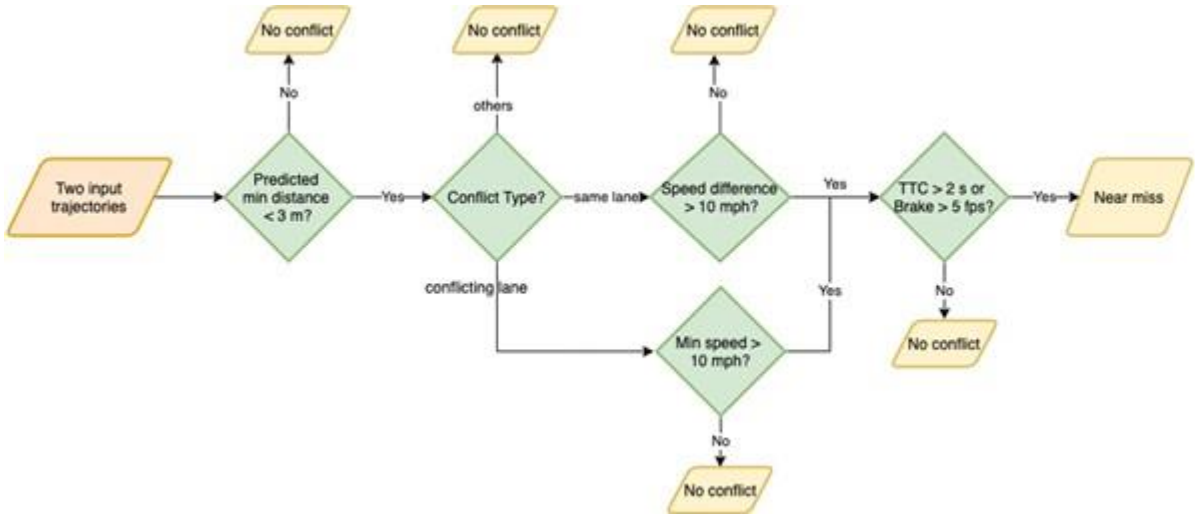


Figure 8-6. Filtering rules for near-misses

8.3.3. Experimental Results

9.3.3.1 Trajectory Prediction

Our intersection trajectory dataset is obtained from three busy intersections in Gainesville, Florida: W University Ave at Tower Rd; Museum Rd at Newell Dr, and W University Ave at 13th St.

We clean the dataset using rule-based filtering. We impose speed limits within the intersection and non-self-intersecting constraints on trajectories. After the above process, we obtain roughly 15,000 trajectories for each intersection, which are then split into training, validation, and testing sets with a roughly 7:1:2

ratio. The validation set is used in model training to avoid overfitting. We refer to the three intersections as Intersection 1, Intersection 2 and Intersection 3.

The quantitative results are shown in Table 8-5. We utilize two metrics: average displacement error (ADE) and final displacement error (FDE) (in meters) to measure the prediction accuracy. The lower the error, the more accurate the approach. We show the prediction errors of different observation and prediction periods. When 20 or 30 observations are given, the model usually gives average displacement errors (ADEs) of less than 1 meter, and final displacement errors (FDEs) of less than 3 meters. In the most extreme case of our settings, when only 10 observations are available and 30 predictions are required, the ADE is about 2 meters, and FDE is about 4–5 meters. The FDE here is relatively high because 10 observations are usually not enough to decide a trajectory's cluster (for example, some lanes can either go straight or turn right, and it's hard to tell a vehicle's intention from only 1 second of data).

Table 8-5. Prediction errors for Intersections 1, 2, and 3.

Observation Length		10			20		30
Prediction Length		10	20	30	10	20	10
Intersection 1	ADE	0.54	1.21	2.09	0.52	1.16	0.61
	FDE	1.02	2.70	4.83	0.97	2.61	1.15
Intersection 2	ADE	0.54	1.23	2.05	0.46	0.91	0.49
	FDE	1.03	2.72	4.29	0.79	1.90	0.82
Intersection 3	ADE	0.66	1.40	2.19	0.58	1.18	0.54
	FDE	1.24	2.89	4.38	1.03	2.44	0.99

9.3.3.2 Near-miss Example Outputs

Here we show a complete example of a predicted near-miss along with the intermediate and final outputs of the pipeline. Figure 8-7 captures a collision between Car 432 and Car 437. After the GO signal was given, Car 432 began turning right when they encountered a pedestrian in their path, which prompted an abrupt stop. Car 437 likely did not anticipate a sudden stop and continued their acceleration until hitting Car 432. Figure 8-8 shows the output of our near-miss prediction module, which foresees the collision. Shown in Figure 8-9, our trajectory prediction algorithm predicts their future motions.

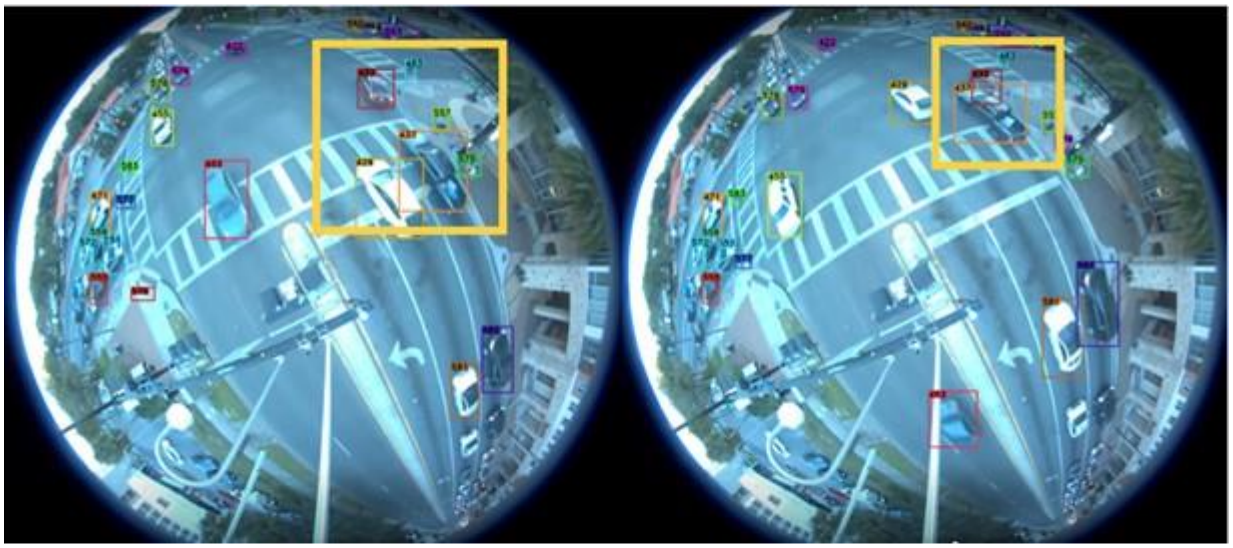
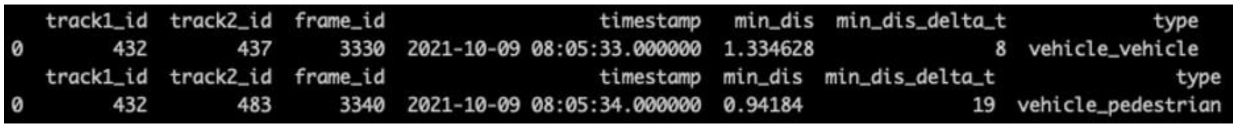


Figure 8-7. Two snapshots of the surveillance video at W University Ave & NW 13th St showing a collision. A collision happens between vehicle 432 and vehicle 437. The left one is before collision, and the right one is when the collision occurs.



```
   track1_id  track2_id  frame_id                   timestamp   min_dis  min_dis_delta_t             type
0        432        437      3330  2021-10-09 08:05:33.000000  1.334628                8  vehicle_vehicle
   track1_id  track2_id  frame_id                   timestamp  min_dis  min_dis_delta_t                type
0        432        483      3340  2021-10-09 08:05:34.000000  0.94184               19  vehicle_pedestrian
```

Figure 8-8. Output of the near-miss prediction module. Notice that both near-miss between Cars 432 and 437 and near-miss between Car 432 and Pedestrian 483 are correctly predicted.

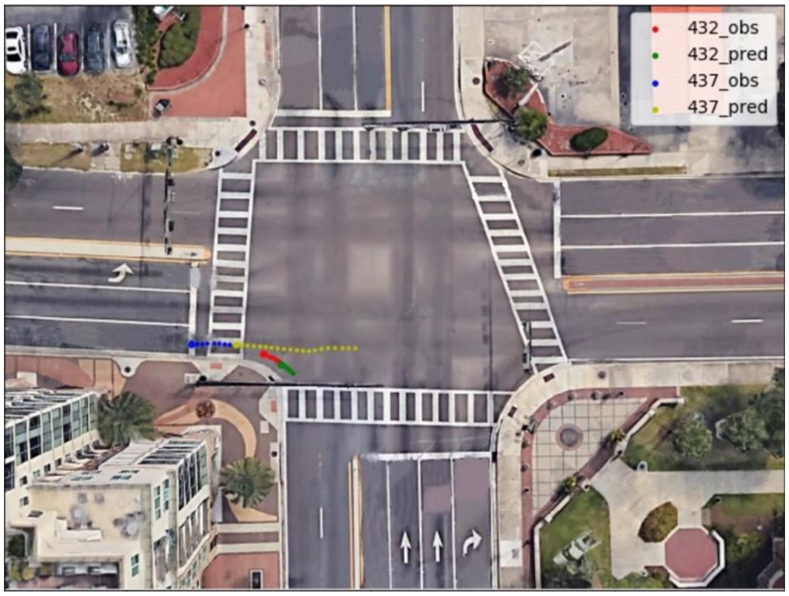


Figure 8-9. Visualizations of the output from trajectory prediction algorithm. As shown in the bottom left corner, the red and blue trajectories represent the observed trajectory of Cars 432 and 437 respectively; the yellow and green ones represent the predicted trajectories.

9.3.3.3 Processing Time and Latency Analysis

The detection and tracking module running on two GPU cores has an inference speed of 35–90 fps, depending on how crowded the intersection objects are. The trajectory prediction and near-miss prediction module running on CPU has an inference speed of 20–30 fps. The message passing time is much shorter compared to the module processing time and can be omitted for latency computation. From end-to-end testing, our full pipeline has a total delay of 0.04 s, which is shorter than the duration of a frame (i.e., our video stream has a frame rate of 10 fps, so that the duration of a frame is 0.1 s). Therefore, our pipeline runs in real time.

8.4. Conclusions

We first described our safety analysis approach for intersections with video and ATSPM data collected at the intersections, and then described how we use similar ideas and a trajectory prediction algorithm to implement real-time near-miss detection. The near-miss prediction module consists of trajectory prediction, pair-wise distance computation, and rule-based filtering. Each module of the pipeline achieves relatively low latency, and we adopt real-time messaging queues for data passing between modules. The system runs in real time, and our system is able to spot near-miss cases before they happen, which verifies the feasibility of such real-time warning systems.

Chapter 9 – Develop Signal Timing Algorithms and Software for Improved Performance at Intersections

9.1. Introduction

With rapid urbanization across the world, the growing volume of vehicular traffic, and increasing complexity of roadway networks has led to problems such as congestion, traffic jams, and traffic incidents. These have been shown to negatively affect productivity and thus the local economy, the well-being of the society, and the environment (Rao and Rao, 2012; Weisbrod et al., 2003; Levy et al., 2010; Zhang and Batterman, 2013). Therefore, keeping traffic flowing smoothly and safely is an important task for traffic authorities.

Thanks to advances in electronics, sensing, computing, data storage, and communications technologies, Intelligent Transportation Systems (ITS) is a fast-growing field. With growing complexity of operating an urban traffic network and with a deluge of data streaming from different sensor modalities, ITS helps traffic engineering practitioners by augmenting their experience with data-driven insights.

Table 9-1. Benefits of acquiring fisheye video data at intersections, over loop-detector-based ATSPM data.

Application	Pre-existing ATSPM	Fisheye Video
Extracting cycle-wise flows, existing signal timing parameters	●	●
Speed, acceleration, deceleration (at light changes)		●
Accurate pedestrian demand, clearance, safety		●
Cycle-wise turning MOVEMENT Counts		●
Phase-wise vehicle composition and lengths (to calibrate simulation with vehicle-type car models and demand composition)		●
Heavy vehicles, bus stoppages (to calibrate simulation with known stops and average stoppage times)		●

Most traffic authorities across the U.S. usually collect high-resolution (10 Hz) loop detector and signal state data. However, there are significant drawbacks, such as being unable to collect precise trajectory information, turn-movement counts, pedestrian walking behaviors, etc. We focus on using data obtained from fisheye lens cameras (Table 9-1). Several new applications involving vehicle and pedestrian trajectories as well as turn-movement counts, vehicle types, etc. are now possible, using the input from video data. Significant cleaning and preprocessing were done to obtain various parameters and metrics. We analyze some of the important ones such as turn-movement counts, pedestrian behaviors, vehicle composition, etc. We briefly discuss traffic simulation frameworks, namely SUMO, i.e. Simulation of

Urban Mobility (Lopez et al., 2018) that allow traffic engineers to model real-world traffic scenarios, study them, and try out improvement strategies in an inexpensive, timely, and safe fashion.

We show that additional data from fisheye lens cameras improve our understanding and modeling of an intersection, over just having ATSPM data. We perform data analytics to show that there is significant variance in terms of turn-movement counts, pedestrian behaviors, vehicle composition, etc. temporally (hour of day, day of week, etc.). We also look for those time periods with similar traffic patterns, which could be served by the same traffic signal timing plan.

Following that, we present a simulation-based approach to customizing signal timing plans based on the traffic behavior at the intersections for various times. We perturb existing signal timing plans and show potential improvements that can be made.

To summarize, the main contributions of this task are listed as follows:

1. Based on video data, we present analysis of specific aspects of traffic behaviors at an intersection, namely, turn-movement counts, pedestrians, and vehicle composition.
2. We develop a simulation basemap of a major intersection of the Trapezium road network (Gainesville, FL) and populate it with traffic, based on arrival patterns from video data.
3. We assess the impact of changing signal timing plans given the simulation model developed above and show potential improvements.

The rest of this report is organized as follows. Section 10.2 gives an overview of the background of the technologies used and SUMO simulation framework. Section 10.3 describes the traffic analytics for the processed video data. Section 10.4 presents the simulation results for improving existing signal timing plans. Finally, Section 10.5 gives a conclusion of our work.

9.2. Background

In this section, we will describe different components that we use for this task. First, we briefly describe the data collection and processing via fisheye lens cameras.

9.3. Fisheye Video

A fisheye lens is a type of camera lens that produces a distortion that allows capture of an ultra-wide-angle field of view, often 100 to 180 degrees. This allows a single camera to capture a much larger scene than regular camera lenses. Such lenses are often paired with video cameras to capture dynamics of traffic intersections (Figure 9-1).

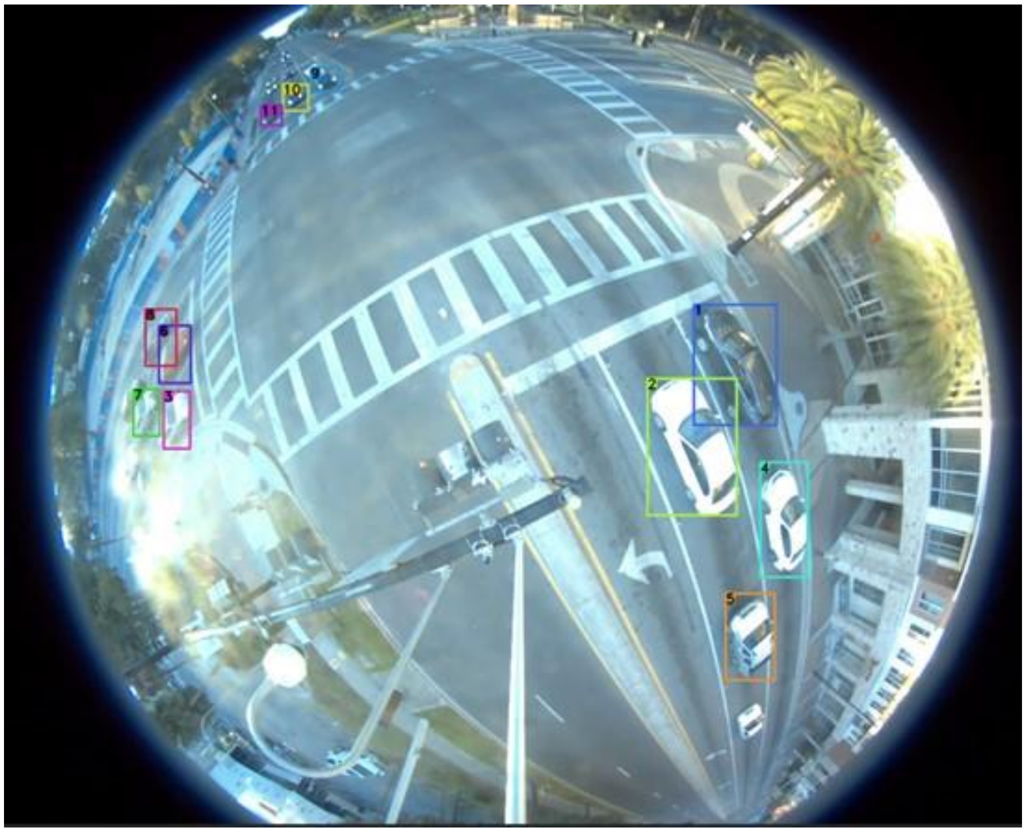


Figure 9-1. A screen capture of the fisheye lens video with bounding boxes overlaid

Video processing algorithms (using deep neural networks architecture, YOLOv4; Bochkovskiy et al., 2020) are used to track individual vehicles across several consecutive frames. This enables us to get trajectory information.

For the purposes of this report, we study Intersection 5060 (Northwest 13th Street and West University Avenue). A single fisheye video camera is mounted 13 feet above the intersection at the southbound approach, on the median (Figure 9-2).

This is a major intersection, forming one of the corners of the Gainesville “Trapezium.” Its northbound and eastbound approaches border the University of Florida campus. There are several residential complexes and commercial establishments (especially restaurants) in the area, giving rise to significant pedestrian traffic.

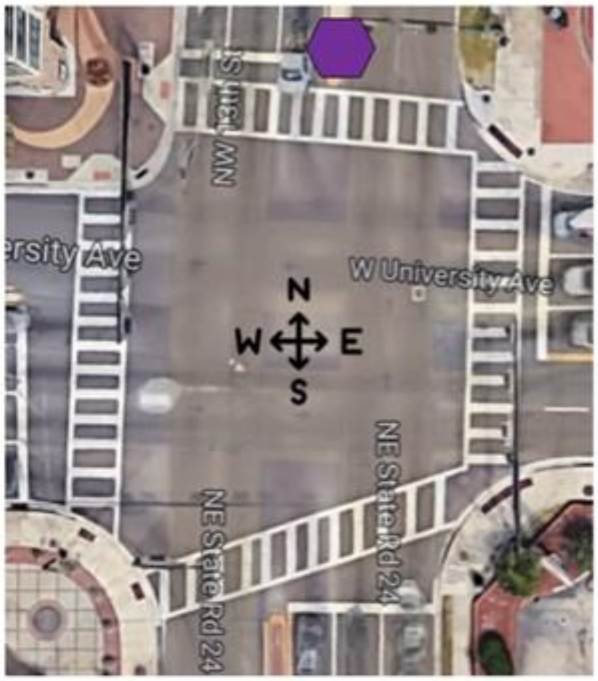


Figure 9-2. Screenshot of the intersection of analysis, W. University Avenue and NW 13th Street. The hexagon at the top-center of the image indicates the location of the fisheye lens camera.

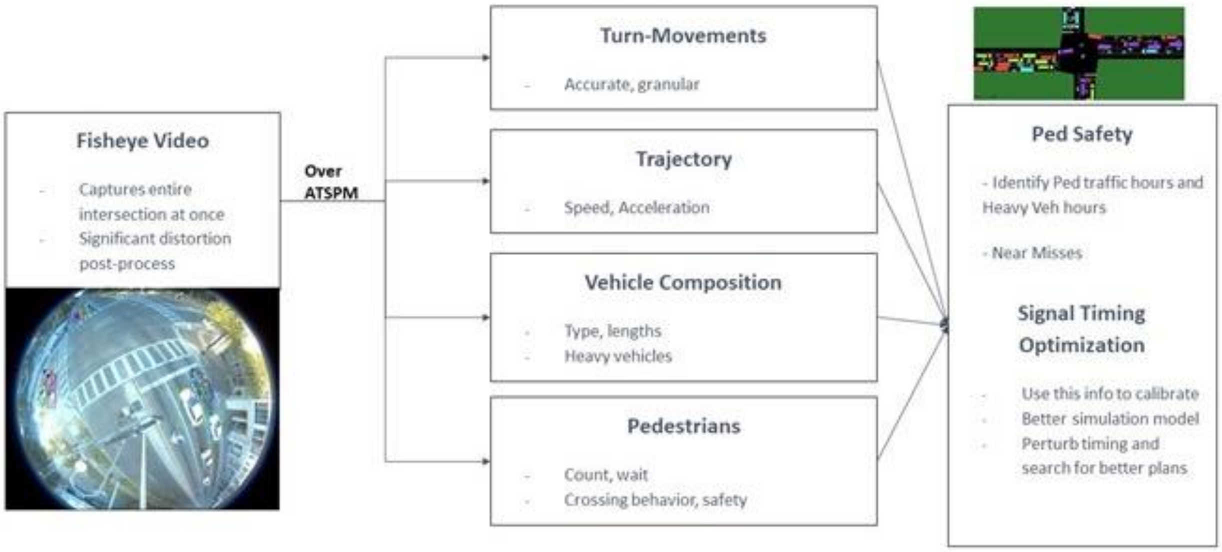

Fisheye Video
- Captures entire intersection at once
- Significant distortion post-process
Over ATSPM
Turn-Movements
- Accurate, granular
Trajectory
- Speed, Acceleration
Vehicle Composition
- Type, lengths
- Heavy vehicles
Pedestrians
- Count, wait
- Crossing behavior, safety
Ped Safety
- Identify Ped traffic hours and Heavy Veh hours
- Near Misses
Signal Timing Optimization
- Use this info to calibrate
- Better simulation model
- Perturb timing and search for better plans

Figure 9-3. Flowchart showing benefits and potential uses of fisheye lens camera data for pedestrian safety and signal timing optimization.

The flowchart above (Figure 9-3) gives a brief overview of how we intend to use fisheye video data for the purposes of (descriptive) temporal analysis and (predictive) signal timing optimization.

9.4. Simulation Framework

We now discuss traffic simulation software, with a focus on SUMO. Traffic simulation frameworks are computational implementations of traffic models with dynamic components (vehicles, traffic signals, pedestrians, etc.) and static components (road geometry, linkages, etc.). A traffic simulation begins with a traffic scenario, which consists of a basemap that defines the static components such as the topology of roads with lanes, junctions that connect these roads, etc. These static components usually do not change their behaviors in the short term (i.e., in seconds or minutes). We add to this basemap, dynamic components that change their states based on predefined behaviors (i.e., cars will change their locations based on their speeds and accelerations, traffic signals will change their lights configuration based on signal plan, etc.). The simulation is started and is allowed to evolve in time. We can thus simulate a variety of basemaps and behaviors and estimate different measures of effectiveness (such as queue lengths and travel times).

Based on the level of detail presented to the user, we can broadly classify traffic simulators as:

- Macroscopic simulators: These simulators simulate high-level flow characteristics such as speed and density by considering the aggregate behaviors of a large collection of vehicles (akin to fluid dynamics). While these simulators allow for computationally inexpensive simulations of large traffic grids, they lack the capability for fine-grained space-time analysis of traffic behaviors.
- Microscopic simulators: These simulators simulate fine-grained vehicle behaviors, often at the individual level. Some may simulate the roadway divided into cells which can be occupied or not occupied by a vehicle, and these cells progressively switch on and off to indicate vehicle movement. Another approach simulates individual vehicles as independent agents that move over roadways and have preprogrammed behaviors such as a car-following model or behavior at intersections. These simulators are usually computationally expensive and are usually used to analyze an intersection or a corridor.
- Mesoscopic simulators: These simulators try and strike a balance between microscopic and macroscopic simulators. Often, they try and analyze groups of vehicles, such as homogenous platoons.

There are several traffic simulators currently in use, such as SUMO VISSIM, CORSIM, AIMSUN, and Paramics.

Simulation of Urban Mobility (SUMO) (Lopez et al., 2018) is a software tool developed by the Institute of Transportation Systems of DLR, the National Aeronautics and Space Research Center of the Federal Republic of Germany in Berlin. SUMO is an open-source (licensed under the GPL), highly portable, microscopic, and continuous road traffic simulation package designed to handle large road networks. SUMO uses its own file formats for traffic networks, but it is able to import files encoded in other popular formats like OpenStreetMap or VISSIM. SUMO is implemented in C++ and uses only portable libraries, thus making it lightweight and fast. SUMO is single-threaded, i.e., uses only one CPU core, but several parallel SUMO processes can be spawned, allowing for parallel simulations.

Signal timing plans of traffic lights can be imported or generated automatically by SUMO. Different phase sequences and phase durations can be implemented. OpenStreetMap can be used to import real-world maps.

The SUMO package consists of a suite of several different interrelated programs:

- SUMO: The heart of the package. It is a microscopic simulation engine with no visualization.
- SUMO-GUI: Visualization tool for SUMO that shows the map and the vehicles moving. Also has a graphical user interface to interact with SUMO.
- NETCONVERT: Network importer and generator. It reads road networks in different formats and converts them into the native SUMO format.
- NETEDIT: A graphical network editor that allows for the editing of maps by hand.
- DUAROUTER: Computes fastest routes through the network and performs the DUA (Dynamic User Assignment).
- JTRROUTER: Computes routes using given junction turning percentages
- DFROUTER: Computes routes based on given induction loop detector measurements
- OD2TRIPS: Decomposes origin-destination (O-D) matrices into single-vehicle trips
- TraCI: TraCI is a package that ships with SUMO and is short for "Traffic Control Interface." It gives access to a user-written Python code (or an external Python program) to a road traffic simulation running in SUMO. It allows for the retrieval of values of simulated objects and the manipulation of their behavior while the simulation is running. TraCI uses a TCP based client/server architecture to provide access to SUMO.

9.5. Traffic Analysis of Fisheye Data

In this section, we will describe our efforts in analyzing the fisheye camera data given the time of day and day of week. We will look for both variation and similarities. Significant variations in different aspects of traffic patterns will indicate to us that customized signal timing plans may be required. The similarities across such sections will indicate to us which signal timing plan can be effectively applied in the future.

9.6. Turn-Movement Count Analysis

One of the key pieces of information that fisheye cameras provide is turn-movement counts, i.e., the number of vehicles that travelled in a certain direction. This information is not easily obtained from loop-detector based ATSPM data, as exit detectors are usually not installed. It is seen that most intersections often have lanes that feed into multiple lanes, e.g., the right-most lane often allows for both through and right-turning traffic. Fisheye lens tracking allows for counting such vehicles. In this section, we will analyze vehicle turn-movements at an hourly level of aggregation.

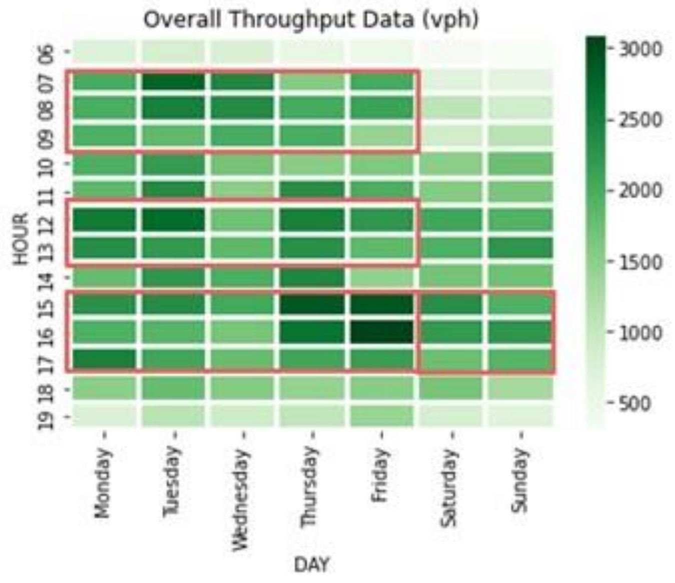


Figure 9-4. Heatmap showing overall intersection average throughput.

We can see above in Figure 9-4 that the overall intersection average throughput (in veh/hour) varies over the course of the day as well as over the day of the week. We can notice weekday AM and PM peaks as well as lower traffic on weekend mornings, which gradually increases in the evening.

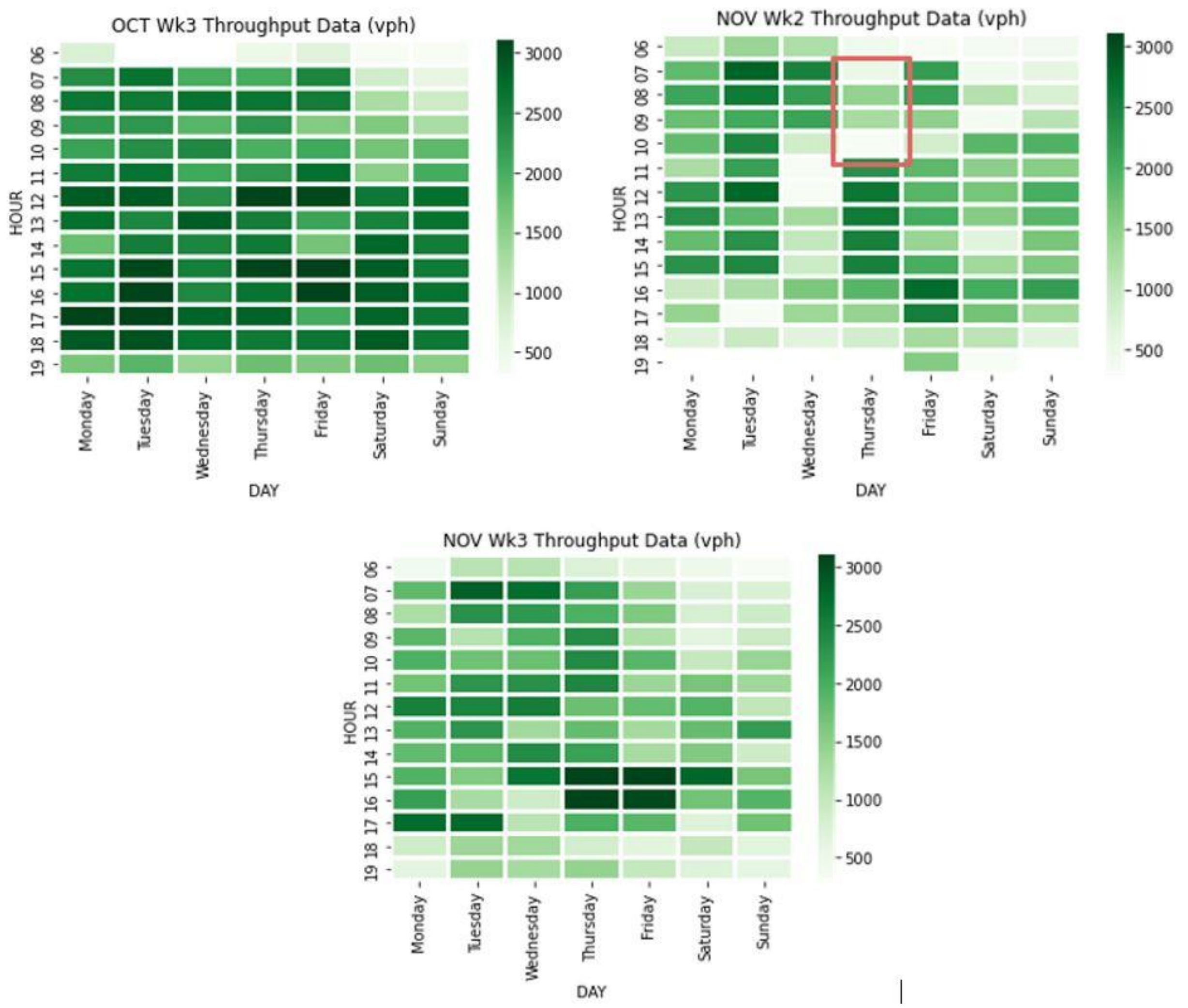


Figure 9-5. Heatmaps showing weekly intersection average throughput.

The data can also be visualized on a weekly basis as shown in Figure 9-5. We can see both variations across the weeks but also consistent trends, especially with respect to AM and PM peaks. The second Thursday of November was Veterans Day, and a morning peak was not seen.

However, the introduction of fisheye lens cameras with tracking algorithms allows us to analyze the data based on turn-movement counts. Below, we present turn-based heatmaps of major and minor movements.

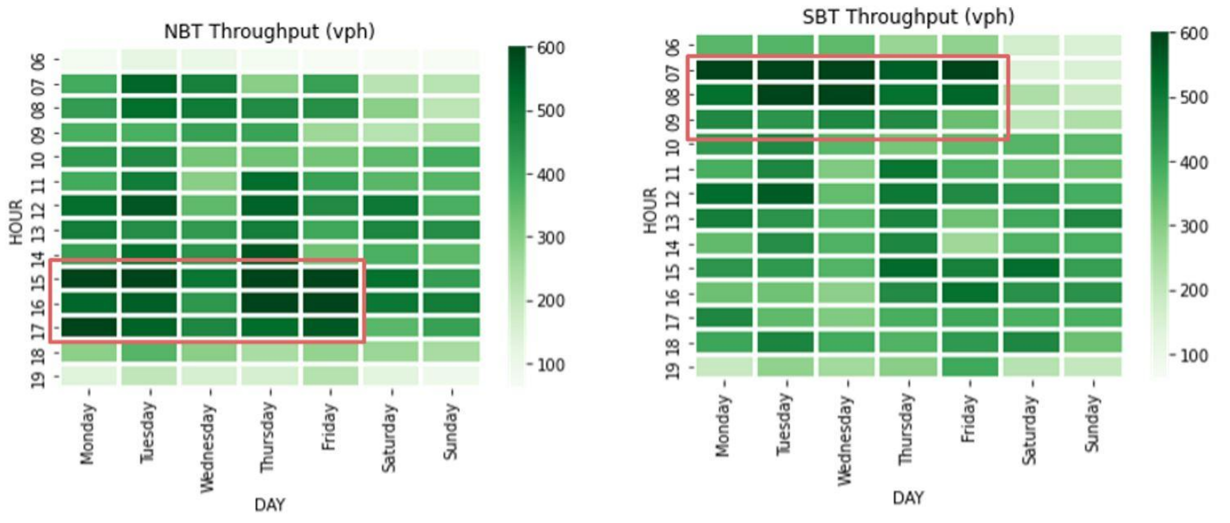


Figure 9-6. Heatmaps showing average throughput along the major traffic directions, i.e., Phase 2/6.

We can see (Figure 9-6) that NBT has a larger PM peak whereas SBT has a larger AM peak. This is likely due to commuters entering the University of Florida campus in the mornings and leaving in the evening.

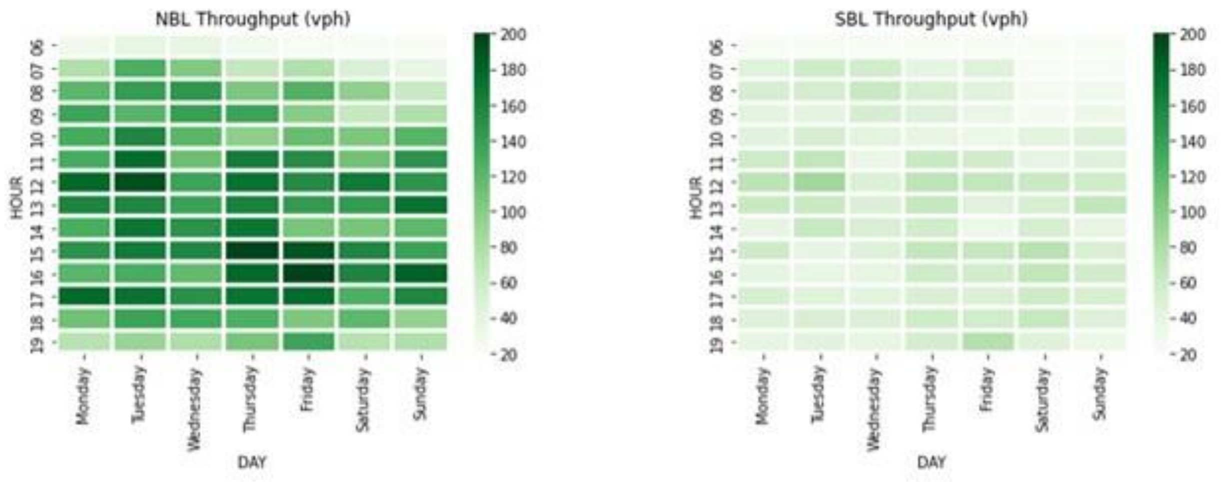


Figure 9-7. Heatmaps showing average throughput along the minor traffic directions, i.e., Phase 4/8.

We can see (Figure 9-7) that both WBT and EBT have significant AM and PM peaks. This is likely due to the fact that while the University of Florida campus is to the west of this intersection. Gainesville Midtown and Downtown areas (with a significant number of commercial establishments) are to the east of this intersection.

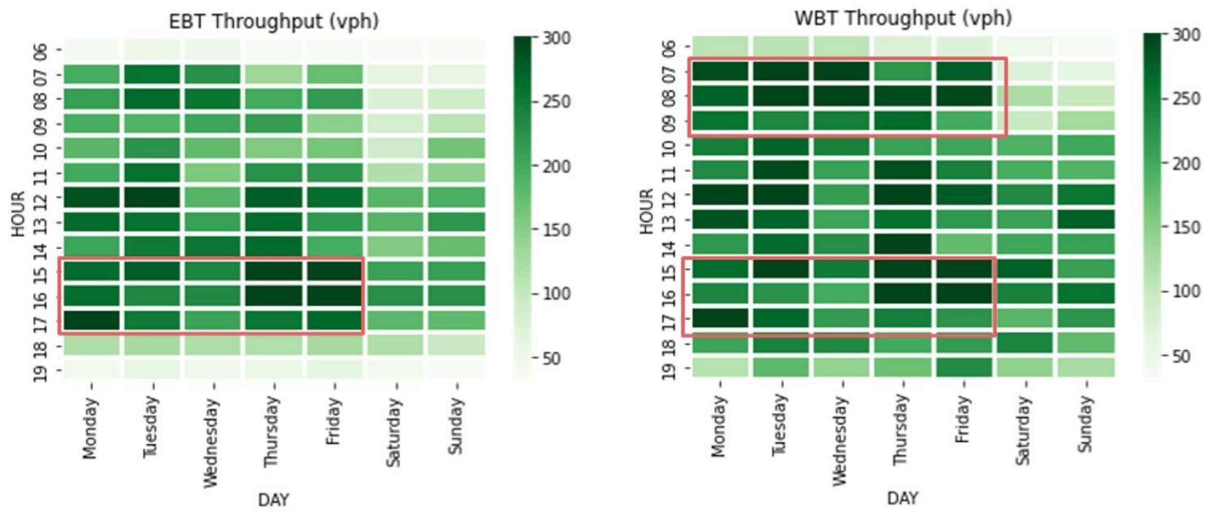


Figure 9-8. Heatmaps showing average throughput along the major left-turn directions.

We see that (Figure 9-8) there are relatively fewer people who take a left-turn towards Downtown when approaching from the southbound direction.

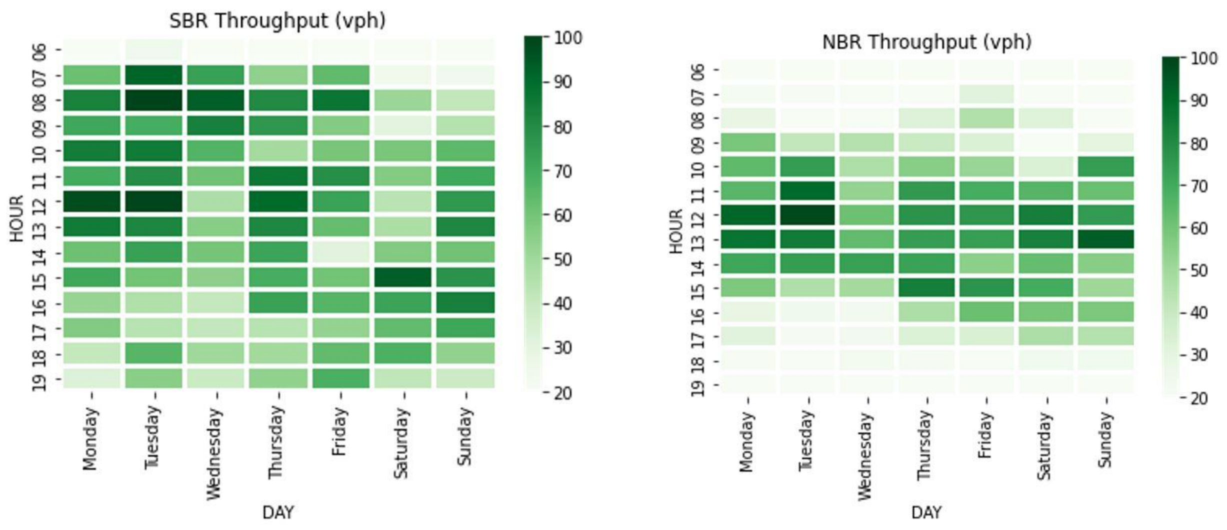


Figure 9-9. Heatmaps showing average throughput along the major right-turn directions.

We can see (Figure 9-9) that overall, more people take right turns into West University Avenue, while fewer people take right turns towards Downtown from this intersection.

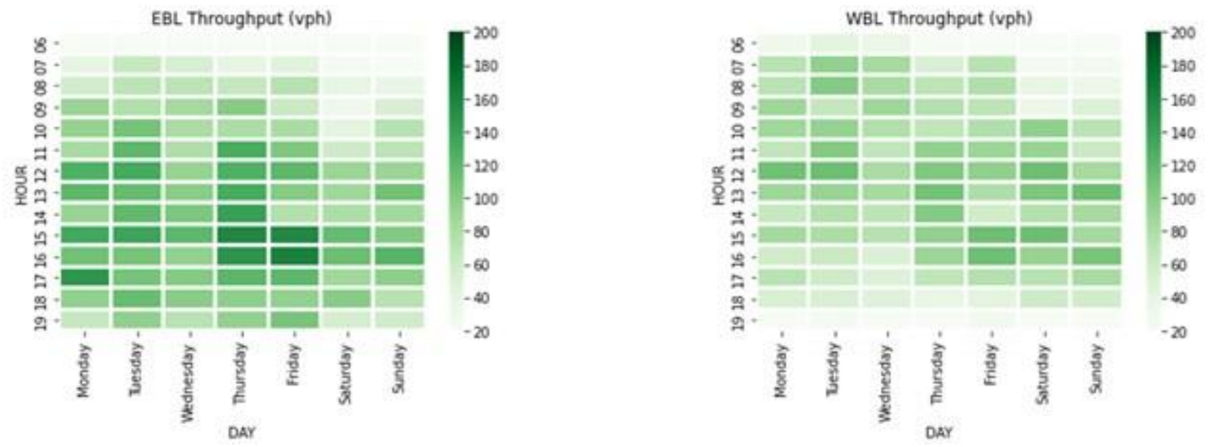


Figure 9-10. Heatmaps showing average throughput along the minor left-turn directions.

We can see (Figure 9-10) that generally fewer people wish to take left turns and get off West University Avenue.

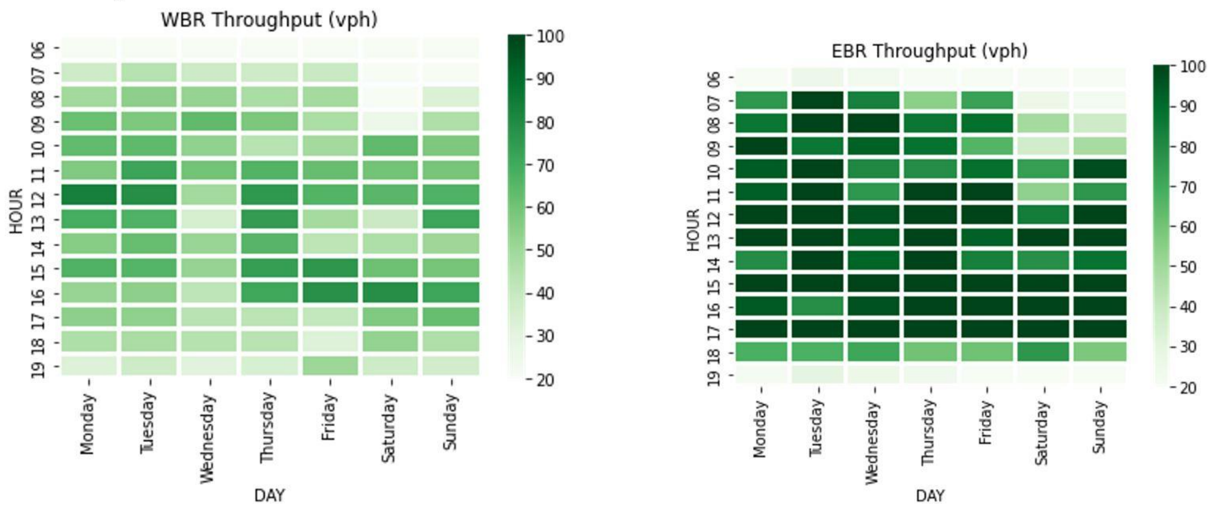


Figure 9-11. Heatmaps showing average throughput along the minor right-turn directions

We can see (Figure 9-11) a lot more people wish to take a right along the eastbound direction, i.e., towards the south along 13th Street.

9.7. Heavy Vehicle Analysis

Fisheye video also allows us to detect heavy motor vehicles, i.e., large vehicles such as delivery trucks and buses. Such vehicles were initially manually annotated by visually observing and labeling them.

Then the YOLOv4 (Bochkovskiy et al., 2020) Deep Learning Network implicitly generated the criteria based on these annotations, and automatically classified them.

Given that this intersection borders the university and has several shopping and eating locations nearby, these large vehicles may pose a safety concern for pedestrians. Figure 9-12 shows the overall average heavy vehicle throughput. It may also be possible to use this information to adjust signal timing plans.

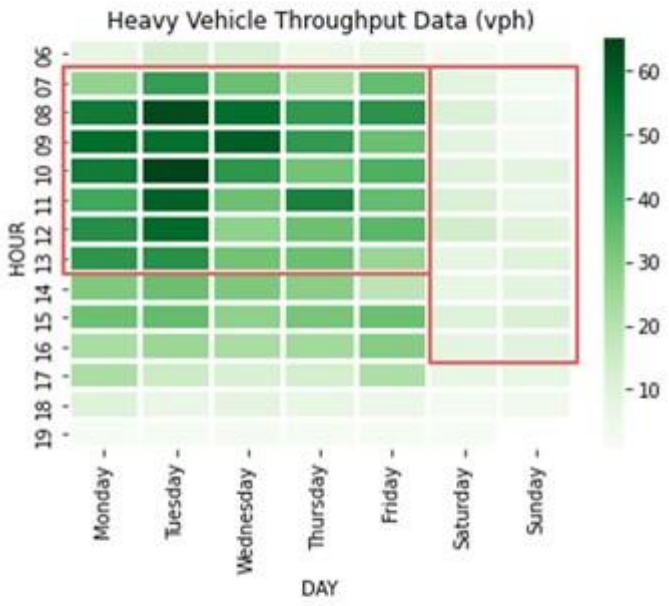


Figure 9-12. Heatmaps showing overall average heavy vehicle throughput.

We can break up the overall chart by weeks as shown in Figure 9-13. We see a persistent trend of large vehicles on weekday mornings and afternoons, while the weekends are largely devoid of them.

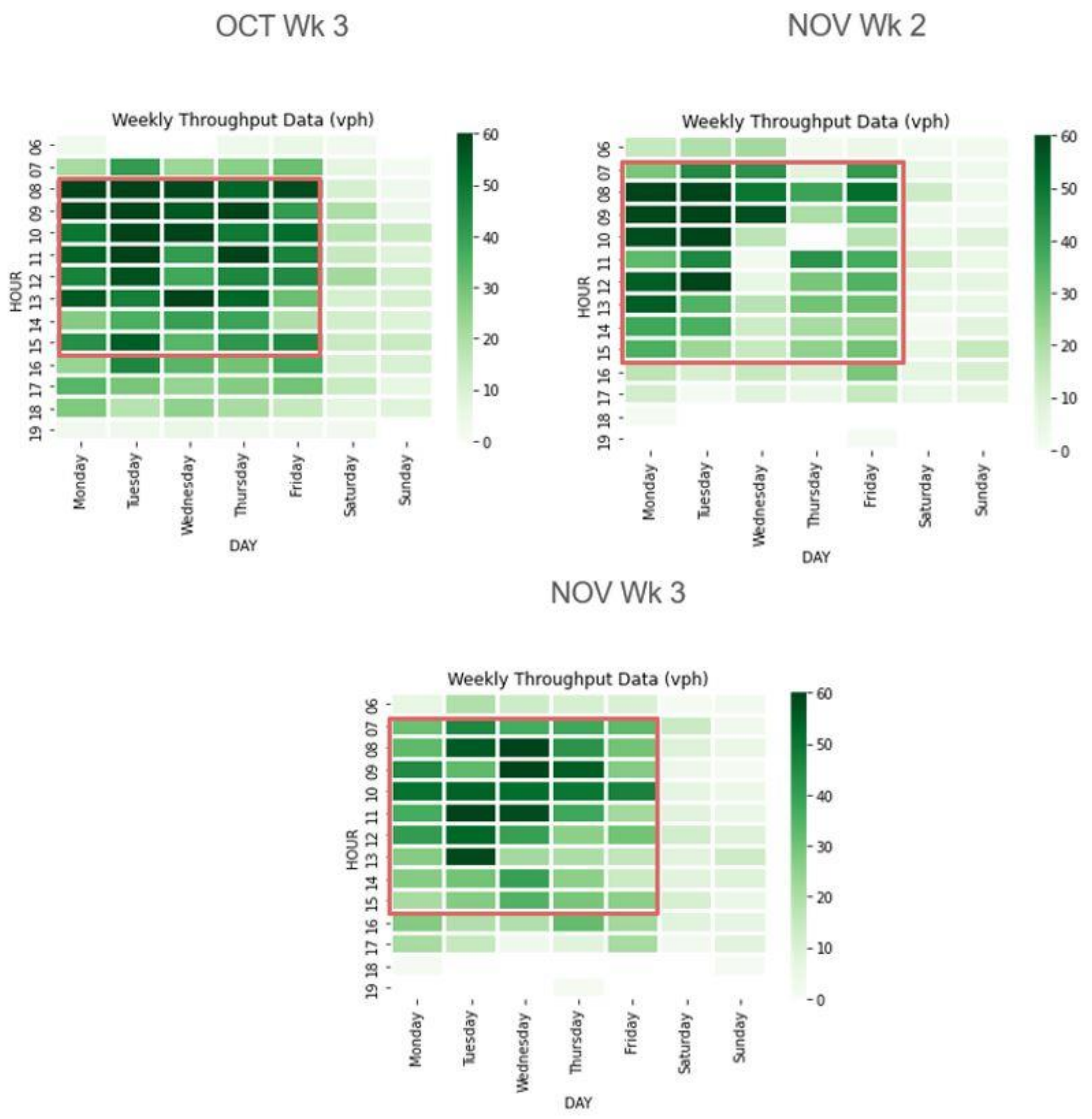


Figure 9-13. Heavy vehicle hourly throughput for three weeks.

9.8. Pedestrian Analysis

Fisheye lens cameras allow us to track individual pedestrians. This allows us to estimate the number of pedestrians crossing along various directions. This contrasts with ATSPM data that simply tracks the number of pedestrian calls but has no way of estimating the actual number of pedestrians.

We can see from Figure 9-14 that the number of pedestrians crossing along the major phase (i.e., north-south) and minor phase (i.e., east-west) to be higher during weekday afternoons. Figure 9-15 shows a similar study across weeks.

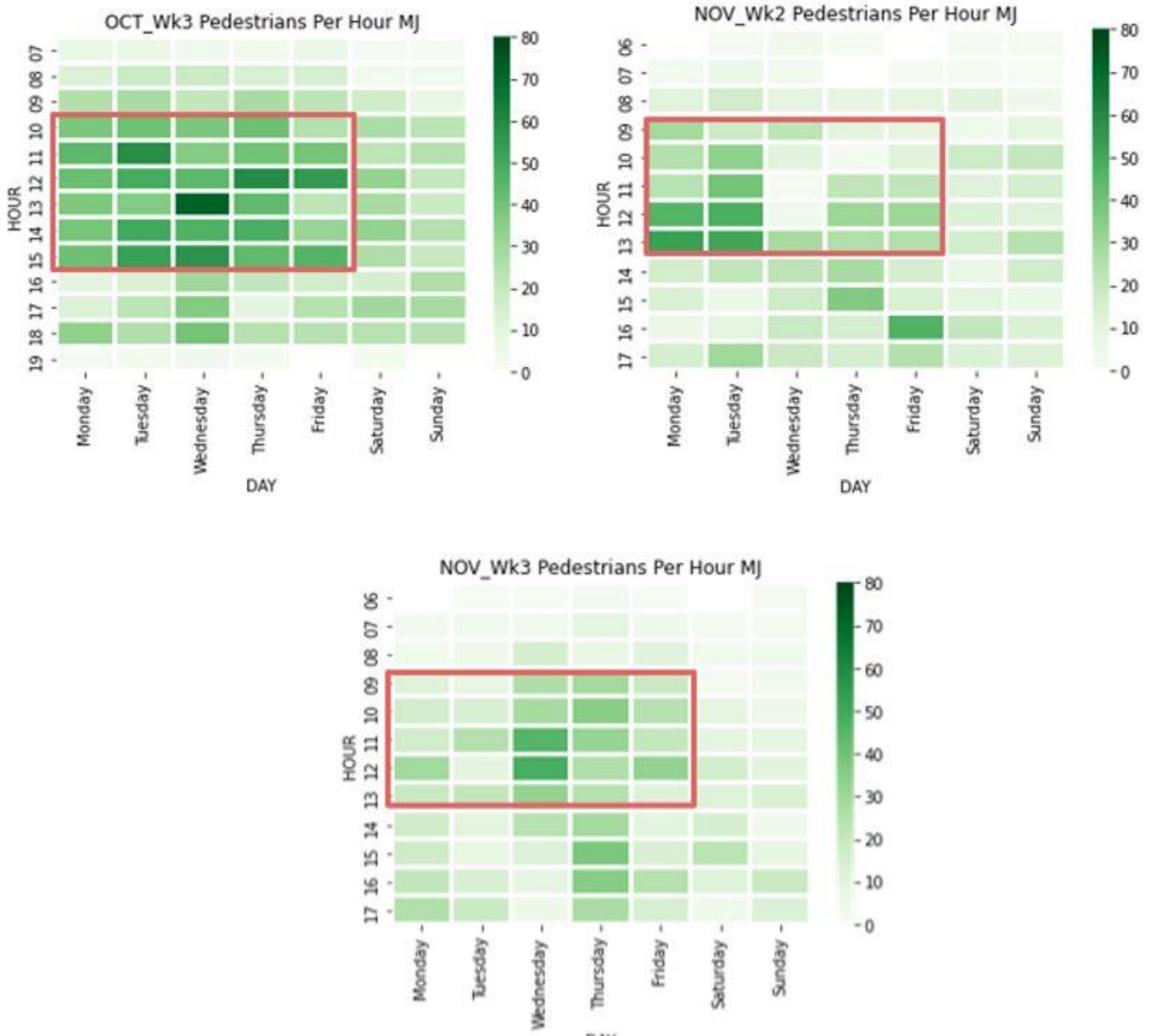
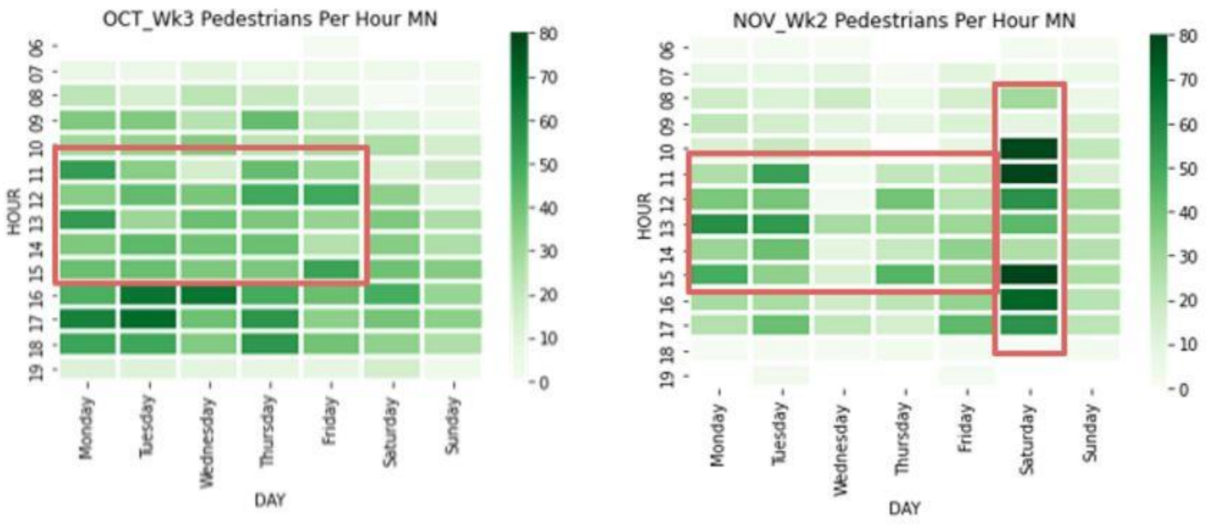


Figure 9-14. Hourly pedestrian throughput along the major traffic flow direction for three weeks.



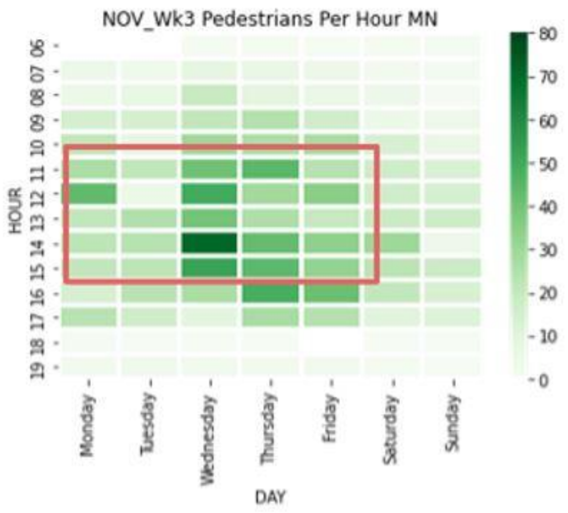


Figure 9-15. Hourly pedestrian throughput along the minor traffic flow direction for three weeks..

An unusual peak was seen on Saturday in the second week of November. This can be explained due to an important football game that day.

We can also estimate the number of cycles impacted by a pedestrian call. This information is shown in Figures 9-16 and 9-17. We see that during the peak pedestrian times, i.e., midday on weekdays, that most cycles have a pedestrian call, both along the major and minor directions.

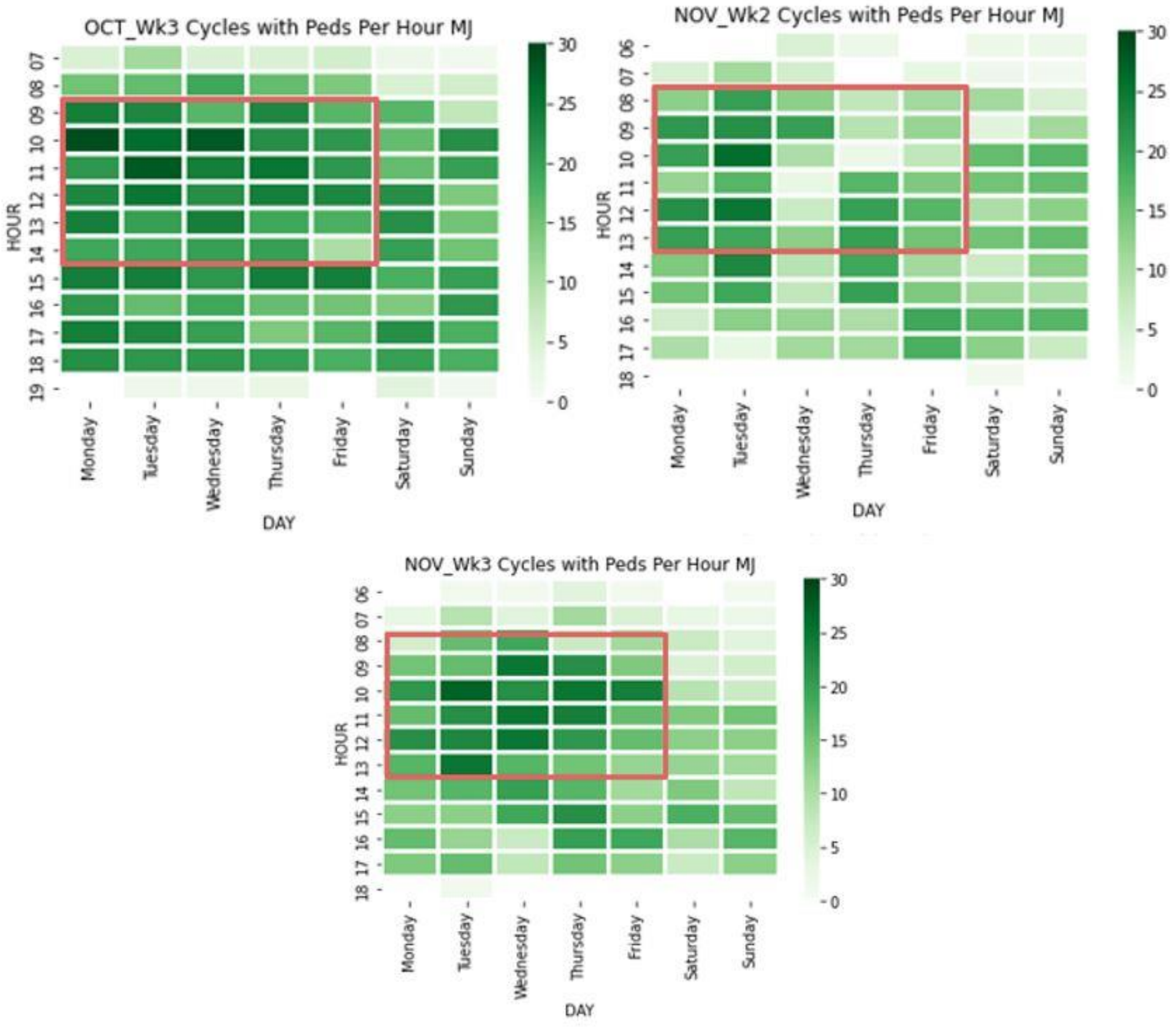


Figure 9-16. Number of cycles impacted by pedestrian calls along the major traffic direction.

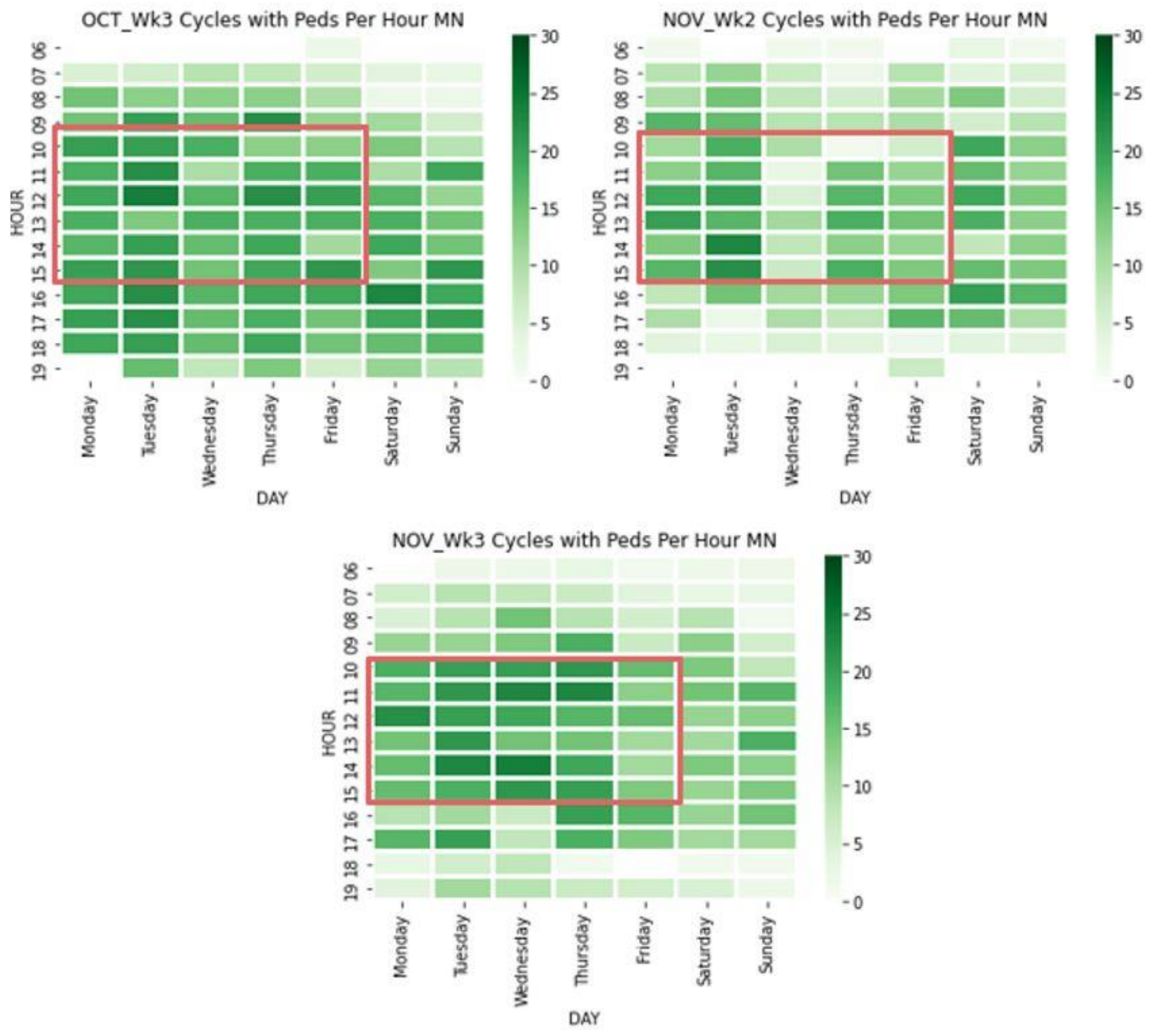


Figure 9-17. Number of cycles impacted by pedestrian calls along the minor traffic direction.

Thus, in this section, we have seen the usefulness of fisheye lens camera videos in providing us descriptive statistics of various road users, including heavy vehicles and pedestrians. Such analysis can lead to better policy decisions, especially with respect to pedestrian safety:

- Restricting right turns and permissive lefts during heavy pedestrian traffic
- Minimizing interaction of heavy vehicles and pedestrians.

9.9. Signal Timing Optimization

In this section, we present our efforts in utilizing the fisheye lens camera data for modeling the intersection dynamics in simulation. With this, we can then vary various signal timing parameters, with the aim of improving performance.

We build a simulation of the intersection we have chosen as shown in Figure 9-18. We then input flows by cycle with the correct turn-movement counts. We also replicate the effect of pedestrian calls within the ring-and-barrier actuated signal timing plan we have implemented in the simulation. We can also use bounding box information to further calibrate vehicle lengths.

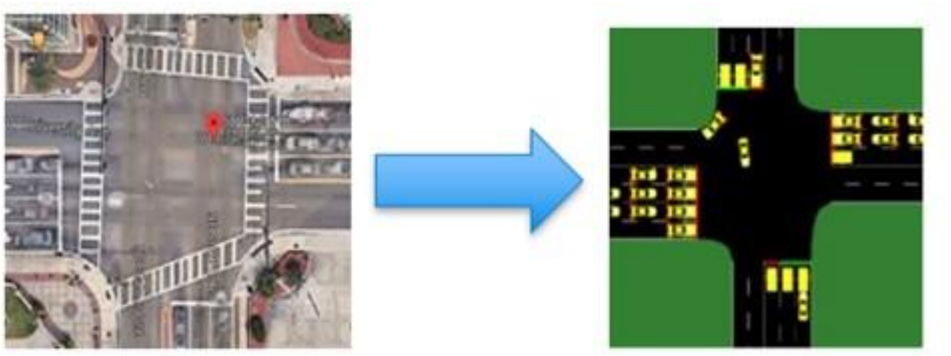


Figure 9-18. Replication of intersection geometry and flows in SUMO simulation.

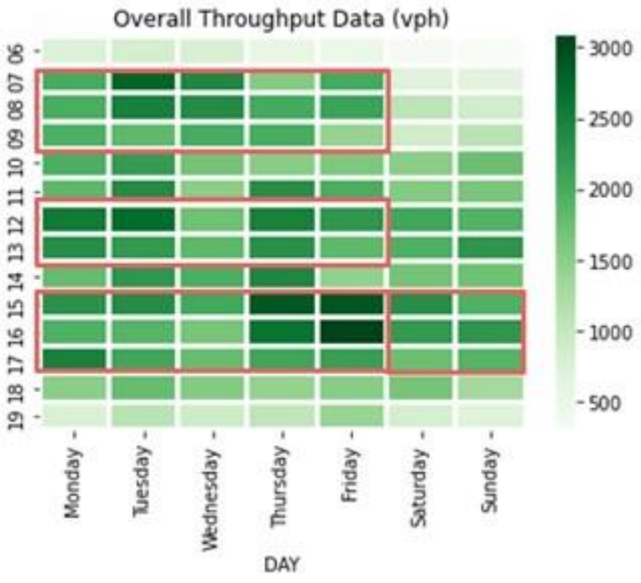


Figure 9-19. Overall traffic throughput shows 4 high-volume times for analysis: weekday AM peak, weekday midday peak, weekday PM peak, and weekday PM peak.

Our earlier analysis in the previous section, leads us to identify 4 important traffic patterns (see Figure 9-19):

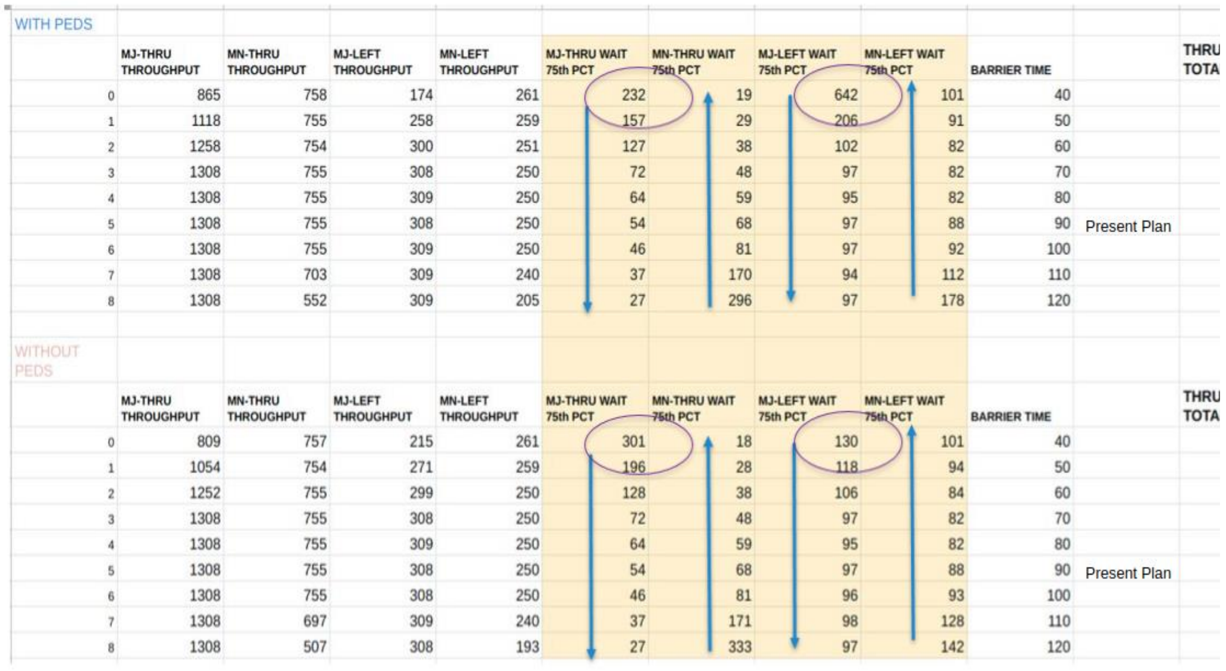
- Weekday AM peak,
- Weekday midday peak

- Weekday PM peak
- Weekday PM peak.

We simulate each of these scenarios with their respective signal timing plans, vehicle flow, and pedestrian calls. We perturb the barrier times and see the effect on the trade-off between the major and minor wait times.

We first see the impact of introducing pedestrian calls, and ignoring them. We can see a clear impact of including the pedestrian calls. Pedestrians require significant time to safely cross the road (here, 30 seconds). Given the ring-and-barrier configuration shown, including pedestrian calls within a cycle, ensures lower time for the left-turning phases that precede them (on both major and minor directions).

Table 9-2. Given the ring-and-barrier scheme at this intersection, left turns are affected by the inclusion of pedestrian calls.

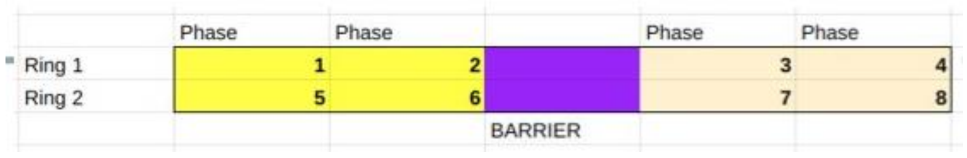


WITH PEDS

	MJ-THRU THROUGHPUT	MN-THRU THROUGHPUT	MJ-LEFT THROUGHPUT	MN-LEFT THROUGHPUT	MJ-THRU WAIT 75th PCT	MN-THRU WAIT 75th PCT	MJ-LEFT WAIT 75th PCT	MN-LEFT WAIT 75th PCT	BARRIER TIME		THRU TOTA
0	865	758	174	261	232	19	642	101	40		
1	1118	755	258	259	157	29	206	91	50		
2	1258	754	300	251	127	38	102	82	60		
3	1308	755	308	250	72	48	97	82	70		
4	1308	755	309	250	64	59	95	82	80		
5	1308	755	308	250	54	68	97	88	90	Present Plan	
6	1308	755	309	250	46	81	97	92	100		
7	1308	703	309	240	37	170	94	112	110		
8	1308	552	309	205	27	296	97	178	120		

WITHOUT PEDS

	MJ-THRU THROUGHPUT	MN-THRU THROUGHPUT	MJ-LEFT THROUGHPUT	MN-LEFT THROUGHPUT	MJ-THRU WAIT 75th PCT	MN-THRU WAIT 75th PCT	MJ-LEFT WAIT 75th PCT	MN-LEFT WAIT 75th PCT	BARRIER TIME		THRU TOTA
0	809	757	215	261	301	18	130	101	40		
1	1054	754	271	259	196	28	118	94	50		
2	1252	755	299	250	128	38	106	84	60		
3	1308	755	308	250	72	48	97	82	70		
4	1308	755	309	250	64	59	95	82	80		
5	1308	755	308	250	54	68	97	88	90	Present Plan	
6	1308	755	308	250	46	81	96	93	100		
7	1308	697	309	240	37	171	98	128	110		
8	1308	507	308	193	27	333	97	142	120		



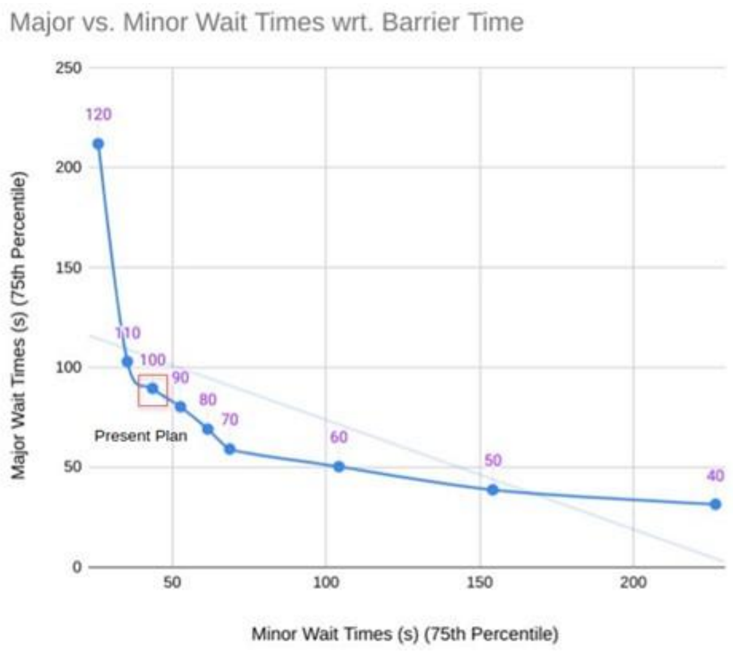
	Phase	Phase		Phase	Phase
Ring 1	1	2		3	4
Ring 2	5	6		7	8
			BARRIER		

We can see from the Table 9-2 above that because this non-negotiable amount of time (here, 30 seconds) must be provided to serve the pedestrian calls, barrier times that previously had acceptable wait times for left-turning traffic now are unacceptable. Hence, it is important to include them while analyzing the trade-off between major and minor street wait times and these barrier times can no longer be considered viable when pedestrian calls are considered. Thus, the inclusion of pedestrian calls is vital during simulation.

Below we present the simulation results for the key traffic times we identified. The plot on the left shows the trade-off for (75th percentile) wait times of major and minor streets. The plot on the right shows the change in throughput when the barrier time is changed.

9.10. Simulations Results: Weekday AM Peak

In this scenario (Figure 9-20), we can see that major wait times could be decreased by lowering the barrier time to 70 or 80 seconds, with relatively small increase in minor wait times. The throughput would remain unchanged. Beyond 70 seconds, there would be a massive increase in minor wait times, with no appreciable improvement in major wait times.



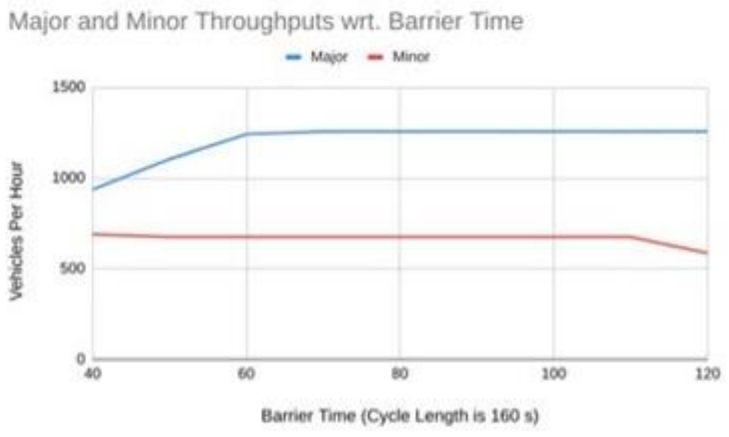


Figure 9-20. Simulations results for weekday AM peak.

9.11. Simulations Results: Weekday Midday Peak

In this scenario (Figure 9-21), we can see that major wait times could be decreased by lowering the barrier time to 70 or 80 seconds, with relatively small increase in minor wait times. The throughput would remain unchanged. Beyond 70 seconds, there would be a massive increase in minor wait times, with no appreciable improvement in major wait times.

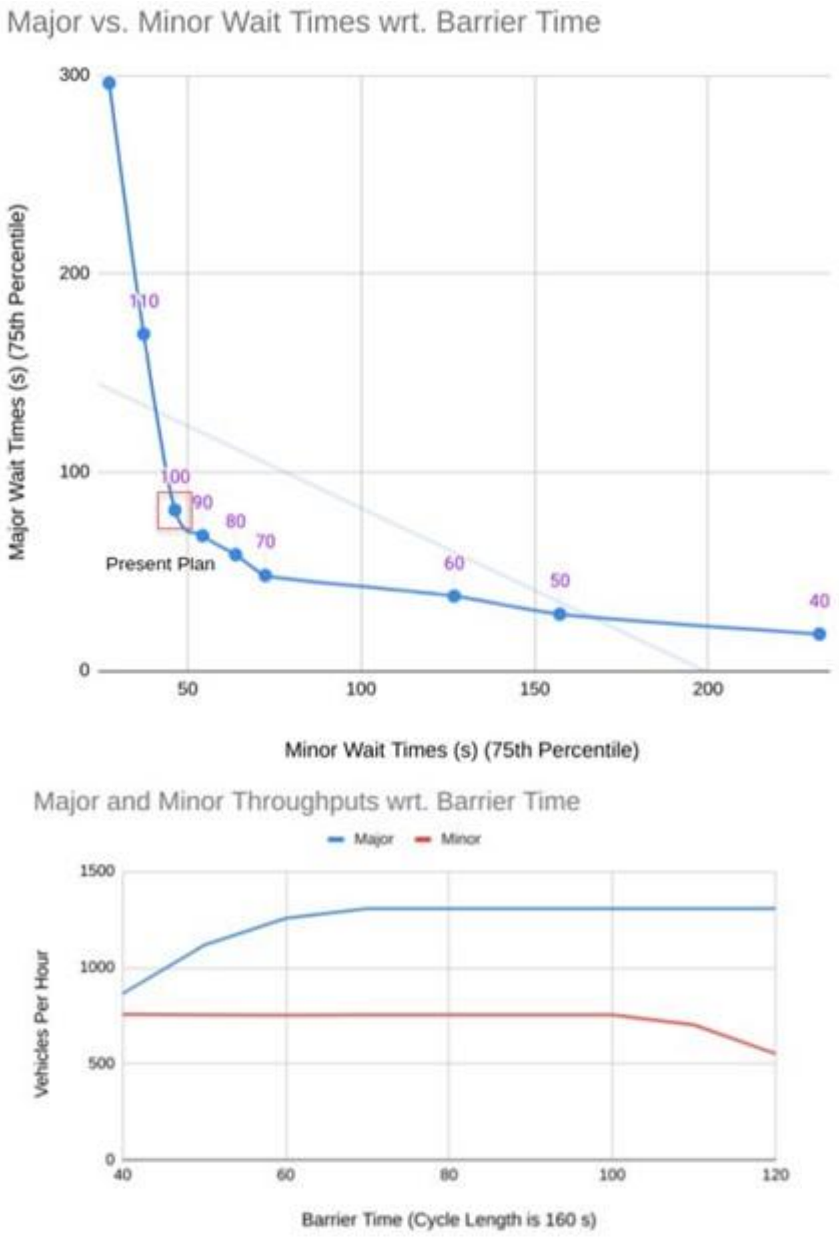
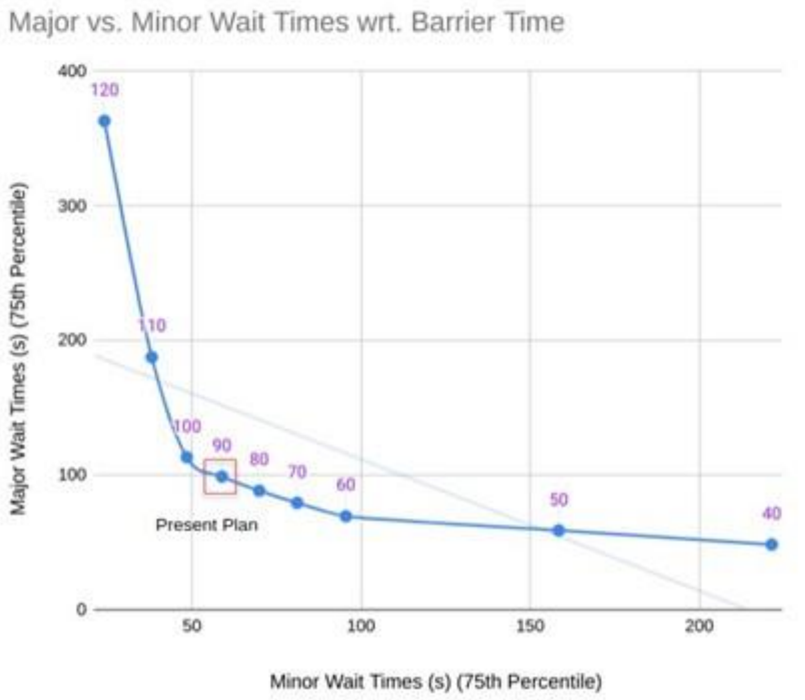


Figure 9-21. Simulations results for weekday midday peak.

9.12. Simulations Results: Weekday PM Peak

In this scenario (Figure 9-22), we can see that major wait times could be decreased slightly by lowering the barrier time to 60 or 70 seconds, but with a more significant increase in minor wait times. The throughput would remain unchanged. Beyond 60 seconds, there would be a massive increase in minor wait times, with no appreciable improvement in major wait times.



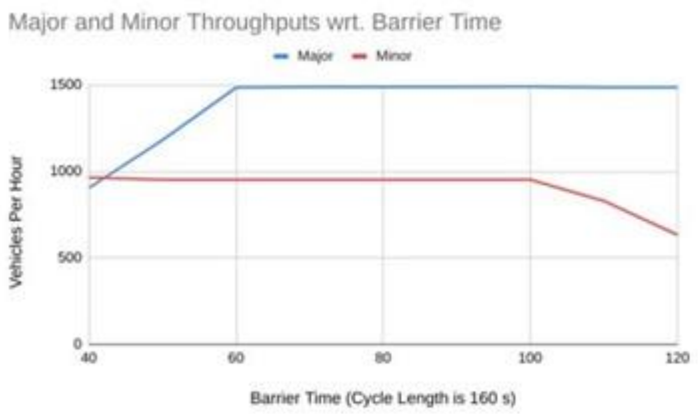
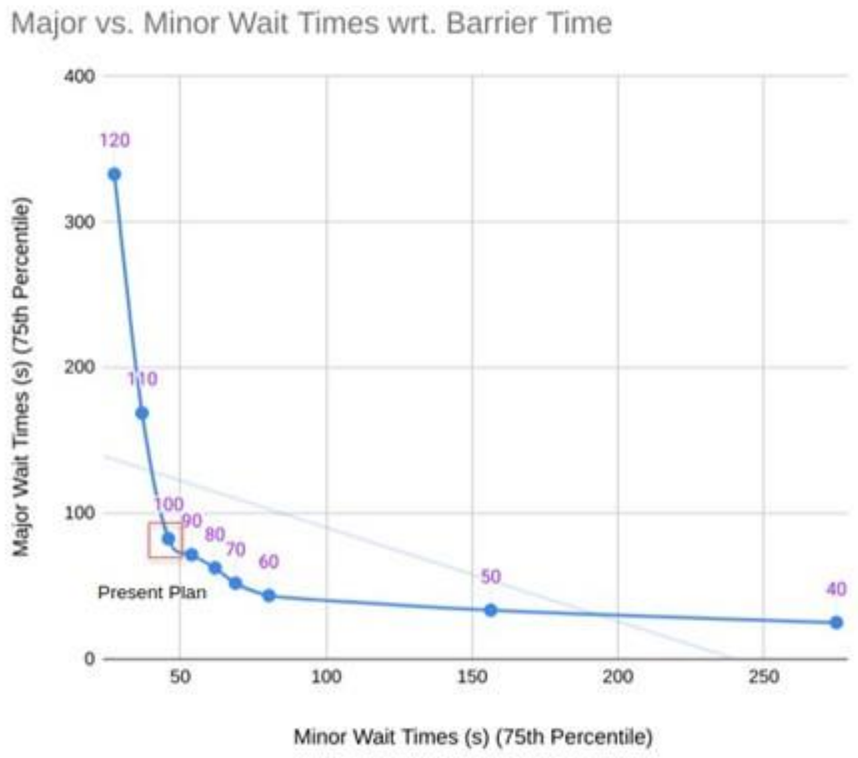


Figure 9-22. Simulations results for weekday PM peak.

9.13. Simulations Results: Weekend PM Peak

In this scenario (Figure 9-23), we can see that major wait times could be decreased by lowering the barrier time to 60 or 70 seconds, but with a more significant increase in minor wait times. The throughput would remain unchanged. Beyond 60 seconds, there would be a massive increase in minor wait times, with no appreciable improvement in major wait times.



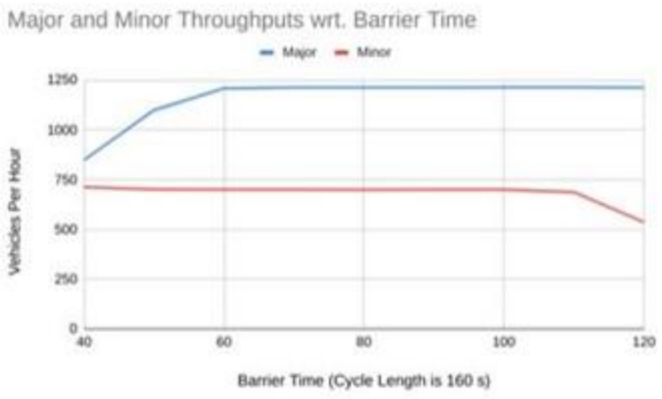


Figure 9-23. Simulations results for weekend PM peak.

9.14. Conclusion

In this report, we have studied the benefits of using fisheye video data for pedestrian safety and simulation.

Fisheye data allow us to improve simulation by:

- Accurate TMCs and flows
- Vehicle length calibration
- Pedestrian information.

We then simulated the intersection of interest with these added inputs. We investigated the difference in outcomes when pedestrians are ignored and included. We find that there is a significant impact, especially on left-turn wait times. We then simulated the intersection for four important high-volume periods in a typical week and obtained the trade-off curve between major and minor waits at various barrier times.

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). “SLIC superpixels compared to state-of-the-art superpixel methods.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

Allen, B. L., Shin, B. T., & Cooper, P. J. (1977) “Analysis of traffic conflicts and collisions.” *Transportation Research Record*, 667:67–74.

Antonini, G., and Thiran, J.-P. (2006). “Counting pedestrians in video sequences using trajectory clustering.” *IEEE Transactions on Circuits and Systems for Video Technology*, 16(8):1008–1020.

Atev, S., Miller, G., and Papanikolopoulos, N. P. (2010). “Clustering of vehicle trajectories.” *IEEE Transactions on Intelligent Transportation Systems*, 11(3):647– 657.

Banerjee, T., Huang, X., Chen, K., Rangarajan, A., and Ranka, S. (2020). “Clustering object trajectories for intersection traffic analysis.” In *Proceedings of the 6th International Conference on Vehicle Technology and Intelligent Transport Systems* (VEHITS 2020). Setúbal, Portugal: Science and Technology Publications, Lda. Pp. 98–105.

Bergmann, P., Meinhardt, T., and Leal-Taixe, L. (2019). “Tracking without bells and whistles.” In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 941–951.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). “Simple online and realtime tracking.” In *2016 IEEE International Conference on Image Processing* (ICIP). New York, NY: IEEE. Pp. 3464–3468.

Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). “YOLOv4: Optimal Speed and Accuracy of Object Detection.” *arXiv*, 2004.10934.

Bookstein, F. L. (1989). “Principal warps: Thin-plate splines and the decomposition of deformations.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6): 567–585.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). “nuScenes: A multimodal dataset for autonomous driving.” In *Proceedings of the Conference on Computer Vision and Pattern Recognition* (CVPR). New York, NY: IEEE. Pp. 11618–11628.

Chang, M.-F., Lambert, J. W., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019). “Argoverse: 3D tracking and forecasting with rich maps.” In *Proceedings of the Conference on Computer Vision and Pattern Recognition* (CVPR). New York, NY: IEEE. Pp. 8748–8757.

Chen, L., Özsu, M. T., and Oria, V. (2005). “Robust and fast similarity search for moving object trajectories.” In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. New York, NY: Association for Computing Machinery (ACM). Pp. 491–502.

Chiu, H.-K., Prioletti, A., Li, J., and Bohg, J. (2020). “Probabilistic 3D multi-object tracking for autonomous driving.” *arXiv*, 2001.05673.

Choy, C., Gwak, J., and Savarese, S. (2019). “4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR). New York, NY: IEEE. Pp. 3070–3079.

Chui, H., and Rangarajan, A. (2003). “A new point matching algorithm for non-rigid registration.” *Computer Vision and Image Understanding*, 89(2–3):114–141.

Cohen, I., Ayache, N., and Sulger, P. (1992). “Tracking points on deformable objects using curvature information.” In *European Conference on Computer Vision*. Berlin, Germany: Springer Nature. Pp. 458–466.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. (2017). “Deformable convolutional networks.” In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 764–773.

Douglas, D.H. and Peucker, T.K. (1973). “Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature.” *The Canadian Cartographer*, 10, 112-122.

Duda, R. O., and Hart, P. E. (1972). “Use of the Hough transformation to detect lines and curves in pictures.” *Communications of the ACM*, 15(1):11–15.

Emami, P., Elefteriadou, L., and Ranka, S. (2021). “Long-range multi-object tracking at traffic intersections on low-power devices.” *IEEE Transactions on Intelligent Transportation Systems*, 23(3):2482–2493.

Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., and Posner, I. (2017). “Vote3deep: Fast object detection in 3D point clouds using efficient convolutional neural networks.” In *2017 IEEE International Conference on Robotics and Automation* (ICRA). New York, NY: IEEE. Pp. 1355–1361.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). “A density-based algorithm for discovering clusters in large spatial databases with noise.” In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Menlo Park, CA: Association for the Advancement of Artificial Intelligence (AAAI). Pp. 226–231.

Fitzgibbon A. W. (2001). “Simultaneous linear estimation of multiple view geometry and lens distortion.” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR 2001), Vol. 1. New York, NY: IEEE. Pp. I-125–I-132.

Ge, W., Collins, R. T., and Ruback, R. B. (2012). “Vision-based analysis of small groups in pedestrian crowds.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):1003–1016.

Geiger, A., Lenz, P., and Urtasun, R. (2012). “Are we ready for autonomous driving? the KITTI vision benchmark suite.” In *Conference on Computer Vision and Pattern Recognition* (CVPR). New York, NY: IEEE. Pp. 3354–3361.

Girshick, R. (2015). “Fast r-cnn.” In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). “Rich feature hierarchies for accurate object detection and semantic segmentation.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 580–587.

Goli, S. A., Far, B. H., and Fapojuwo, A. O. (2018). “Vehicle Trajectory Prediction with Gaussian Process Regression in Connected Vehicle Environment.” In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV)*. New York, NY: IEEE. Pp. 550–555.

Graham, B., Engelcke, M., and van der Maaten, L. (2018). “3D Semantic Segmentation with Submanifold Sparse Convolutional Networks.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 9224–9232.

Hayward, J. C. (1972). “Near miss determination through use of a scale of danger.” *Highway Research Record*, 384:24–34.

He, C., Zeng, H., Huang, J., Hua, X.-S., and Zhang, L. (2020). “Structure aware single-stage 3D object detection from point cloud.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 11873–11882.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916.

Hu, H.-Y., Qu, Z.-W., and Li, Z.-H. (2014). “Multi-level trajectory learning for traffic behavior detection and analysis.” *Journal of the Chinese Institute of Engineers*, 37(8):995–1006.

Huang, X., Yang, C., Ranka, S., Rangarajan A. (2018). “Supervoxel-based segmentation of 3D imagery with optical flow integration for spatiotemporal processing.” *IPSJ Transactions on Computer Vision and Applications*, 10(1): 9

Huang, X., He, P., Rangarajan, A., Ranka, S. (2020). “Intelligent Intersection: Two-Stream Convolutional Networks for Real-time Near-Accident Detection in Traffic Video.” *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 6(2), Article 10, 1-28.

Karunasekera, H., Wang, H., and Zhang, H. (2019). “Multiple object tracking with attention to appearance, structure, motion and size.” *IEEE Access*, 7:104423–104434.

Klokov, R. and Lempitsky, V. (2017). “Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models.” In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 863–872.

Kong, T., Sun, F., Liu, H., Jiang, Y., Li, L., and Shi, J. (2020). “Foveabox: Beyond anchor-based object detection.” *IEEE Transactions on Image Processing*, 29:7389–7398.

Kumar, D., Bezdek, J. C., Rajasegarar, S., Leckie, C., and Palaniswami, M. (2017). “A visual-numeric approach to clustering and anomaly detection for trajectory data.” *The Visual Computer*, 33(3):265–281.

Landrieu, L., and Simonovsky, M. (2018). “Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 4558–4567.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. (2019). "Pointpillars: Fast encoders for object detection from point clouds." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 12697–12705.

Law, H., and Deng, J. (2018). "CornerNet: Detecting objects as paired keypoints." In *Proceedings of the European Conference on Computer Vision* (ECCV). Berlin, Germany: Springer Nature. Pp. 734–75

Levy, J. I., Buonocore, J. J., and Stackelberg. (2010). "Evaluation of the public health impacts of traffic congestion: a health risk assessment." *Environmental Health*, 9(1), art. 65.

Li, E., Wang, S., Li, C., Li, D., Wu, X., and Hao, Q. (2020). "SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System." In *2020 IEEE Intelligent Vehicles Symposium* (IV). New York, NY: IEEE. Pp. 837–844.

Li, J., Chen, B. M., and Hee Lee, G. (2018). "SO-Net: Self-Organizing Network for Point Cloud Analysis." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 9397–9406.

Liu, L., Ouyang, W., Wang, X. et al. Deep Learning for Generic Object Detection: A Survey. *International Journal Computer* Vision 128, 261–318 (2020). https://doi.org/10.1007/s11263-019-01247-4.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). "SSD: Single shot multibox detector." In *European Conference on Computer Vision*. Berlin, Germany: Springer Nature. Pp. 21–37.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wie, E.. (2018). "Microscopic Traffic Simulation using SUMO." In *IEEE Intelligent Transportation Systems Conference* (ITSC). New York, NY: IEEE. Pp. 2575–2582.

Mahmud, S. S., Ferreira, L., Hoque, M. S., & Tavassoli, A. (2017). "Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs." *IATSS Research*, 41(4):153–163.

Morris, B. T., and Trivedi, M. M. (2011). "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2287–2301.

Nadimi, N., Behbahani, H., and Shahbazi, H. (2016). "Calibration and validation of a new time-based surrogate safety measure using fuzzy inference system." *Journal of Traffic and Transportation Engineering*, 3(1):51–58.

Nielsen, F. (2016). *Introduction to HPC with MPI for Data Science*. Berlin, Germany: Springer.

Nikodem, M., Słabicki, M., Surmacz, T., Mrówka, P., and Dołęga, C. (2020). "Multi-camera vehicle tracking using edge computing and low-power communication." *Sensors*, 20(11), art. 3334.

Peppa, M. V., Komar, T., Xiao, W., James, P., Robson, C., Xing, J., and Barr, S. (2021). "Towards an end-to-end framework of CCTV-based urban traffic volume detection and prediction." *Sensors*, 21(2), art. 629.

Pham, Q.-H., Nguyen, T., Hua, B.-S., Roig, G., and Yeung, S.-K. (2019). “JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds with Multi-Task Point- wise Networks and Multi-Value Conditional Random Fields.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR). New York, NY: IEEE. Pp. 8827–8836.

Piciarelli, C., and Foresti, G. L. (2006). “On-line trajectory clustering for anomalous events detection.” *Pattern Recognition Letters*, 27(15):1835–1842.

Piciarelli, C., Micheloni, C., and Foresti, G. L. (2008). “Trajectory-based anomalous event detection.” *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1544–1554.

Prätzlich, T., Driedger, J., and Müller, M. (2016). “Memory-restricted multiscale dynamic time warping.” In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP). New York, NY: IEEE. Pp. 569–573. IEEE.

Qi, C. R., Litany, O., He, K., and Guibas, L. J. (2019). “Deep Hough Voting for 3D Object Detection in Point Clouds.” In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 9277–9286.

Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2018). “Frustum PointNets for 3D Object Detection from RGB-D Data.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 918–927.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 652–660.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.” In *Advances in Neural Information Processing Systems*. New York, NY: Association for Computing Machinery (ACM). Pp. 5099–5108.

Ramer, U. (1972). “An iterative procedure for the polygonal approximation of plane curves.” *Computer Graphics and Image Processing*, 1(3): 244-256.

Rabiner, L. (1993). *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: PTR Prentice Hall.

Rao, A. M., and Rao, K. R. (2012). “Measuring urban traffic congestion: a review.” *International Journal for Traffic & Transport Engineering*, 2(4):286–305.

Redmon, J., and Farhadi, A. (2017). “Yolo9000: better, faster, stronger.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 7263–7271.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). “You only look once: Unified, real-time object detection.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 779–788.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). “You only look once: Unified, real-time object detection.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). “Faster r-CNN: Towards real-time object detection with region proposal networks.” In *Advances in Neural Information Processing Systems*. New York, NY: Association for Computing Machinery (ACM). Pp. 91–99.

Riegler, G., Osman Ulusoy, A., and Geiger, A. (2017). “OctNet: Learning Deep 3D Representations at High Resolutions.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 3577–3586.

Rote, G. (1991). “Computing the minimum Hausdorff distance between two point sets on a line under translation.” *Information Processing Letters*, 38(3):123–127.

Roy, A., Gale, N., and Hong, L. (2011). “Automated traffic surveillance using fusion of doppler radar and video information.” *Mathematical and Computer Modelling*, 54(1-2):531–543.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). “Overfeat: Integrated recognition, localization and detection using convolutional networks.” *arXiv preprint*, 1312.6229.

Shen, Y., Feng, C., Yang, Y., and Tian, D. (2018). “Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling.” In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 4548–4557.

Shi, S., Wang, X., and Li, H. (2019). “PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 770–779.

Silva, D. F., and Batista, G. E. (2016). “Speeding up all-pairwise dynamic time warping matrix calculation.” In *Proceedings of the 2016 SIAM International Conference on Data Mining*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM). Pp. 837–845.

Singleton, P. A., and Runa, F. (2021). “Pedestrian traffic signal data accurately estimates pedestrian crossing volumes.” *Transportation Research Record*, 2675(6):429–440.

Sprengel, R., Rohr, K., and Stiehl, H. S. (1996). “Thin-plate spline approximation for image registration.” in *Proceedings of 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Vol. 3. New York, NY: IEEE. Pp. 1190–1191.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., and Kautz, J. (2018). “SPLATNet: Sparse Lattice Networks for Point Cloud Processing.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 2530–2539.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). “Multi-view Convolutional Neural Networks for 3D Shape Recognition.” In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 945–953.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. (2020). “Scalability in perception for autonomous driving: Waymo open dataset.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 2446–2454.

Tang, Z., Naphade, M., Liu, M-Y., Yang, X., Birchfield, S., Wang, S., Kumar, R., Anastasiu, D., Hwang, J-N.. (2019). "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification." *IEEE Conference on Computer Vision and Pattern Recognition*. 8797–8806.

Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., and Han, S. (2020). "Searching efficient 3D architectures with sparse point-voxel convolution." In *European Conference on Computer Vision*. Berlin, Germany. Pp. 685–702.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). "KPConv: Flexible and Deformable Convolution for Point Clouds." In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 6411–6420.

Tian, Z., Shen, C., Chen, H., and He, T. (2019). "Fcos: Fully convolutional one-stage object detection." In *Proceedings of the IEEE International Conference on Computer Vision*. New York, NY: IEEE. Pp. 9627–9636.

Urbanik, T., Tanaka, A., Lozner, B., Lindstrom, E., Lee, K., Quayle, S., Beaird, S., Tsoi, S., Ryus, P., Gettman, D., et al. (2015). *Signal Timing Manual,* Vol. 1. Washington, D.C.: Transportation Research Board.

Vlachos, M., Kollios, G., and Gunopulos, D. (2002). Discovering similar multidimensional trajectories. In *Proceedings 18th International Conference on Data Engineering.* New York, NY: IEEE. Pp. 673–684.

Wang, X., Tieu, K., and Grimson, E. (2006). "Learning semantic scene models by trajectory analysis." In *European Conference on Computer Vision*. Berlin, Germany: Springer Nature. Pp. 110–123.

Wang, Y., Chen, X., You, Y., Li, L. E., Hariharan, B., Campbell, M., Weinberger, K. Q., and Chao, W.-L. (2020). "Train in Germany, test in the USA: Making 3D object detectors generalize." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 11713–11723.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). "Dynamic Graph CNN for Learning on Point Clouds." *ACM Transactions on Graphics*, 38(5), art. 146.

Weisbrod, G., Vary, D., and Treyz, G. (2003). "Measuring economic costs of urban traffic congestion to business." *Transportation Research Record*, 1839(1):98–106.

Weng, X. and Kitani, K. (2019). "A baseline for 3D multi-object tracking." *arXiv preprint*, 1907.03961.

Wojke, N., Bewley, A., and Paulus, D. (2017). "Simple online and realtime tracking with a deep association metric." In *2017 IEEE International Conference on Image Processing* (ICIP). New York, NY: IEEE. Pp. 3645–3649. IEEE.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). "3D ShapeNets: A Deep Representation for Volumetric Shapes." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 1912–1920.

Yan, Y., Mao, Y., and Li, B. (2018). "SECOND: Sparsely Embedded Convolutional Detection." *Sensors*, 18(10), art. 3337.

Yang, B., Luo, W., and Urtasun, R. (2018). "Pixor: Real-time 3D object detection from point clouds." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 7652–7660.

Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., and Trigoni, N. (2019). "Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds." In *Advances in Neural Information Processing Systems*. New York, NY: Association for Computing Machinery (ACM). Pp. 6737–6746.

Yin, T., Zhou, X., and Krähenbühl, P. (2020). "Center-based 3D Object Detection and Tracking." *arXiv preprint*, 2006.11275.

Yin, T., Zhou, X., and Krähenbühl, P. (2020). "Center-based 3D Object Detection and Tracking." *arXiv preprint*, 2006.11275.

Zhang, K., and Batterman, S. (2013). "Air pollution and health risks due to vehicle traffic." *Science of the Total Environment*, 450:307–316.

Zhang, T., Lu, H., and Li, S. Z. (2009). "Learning semantic scene models by object classification and trajectory clustering." In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 1940–1947. IEEE.

Zhao, H., Sha, J., Zhao, Y., Xi, J., Cui, J., Zha, H., and Shibasaki, R. (2011). "Detection and tracking of moving objects at intersections using a network of laser scanners." *IEEE Transactions on Intelligent Transportation Systems*, 13(2):655–670.

Zhou, X., Wang, D., and Krähenbühl, P. (2019a). "Objects as points." *arXiv preprint*, 1904.07850.

Zhou, X., Zhuo, J., and Krahenbuhl, P. (2019b). "Bottom-up object detection by grouping extreme and center points." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (CVPR). New York, NY: IEEE. Pp. 850–859.

Zhou, Y., and Tuzel, O. (2018). "Voxelnet: End-to-end learning for point cloud based 3D object detection." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE. Pp. 4490–4499.

Appendix A. Machine Learning Terminology and Acronyms

ATSPMs	Automated traffic signal performance measures, included in the Every Day Counts 4 technology initiative, comprise a suite of performance measures, data collection, and data analysis tools to support objectives and performance-based approaches to traffic signal operations, maintenance, management, and design to improve the safety, mobility, and efficiency of signalized intersections for all users.
CCTV	Closed-circuit television
CNN or **ConvNe**t	– Convolutional neural network
CONVs	Convolutional neural network layers
DSRC	Dedicated short-range communications
FCs	Fully connected layers
GPU	Graphics processing unit – a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device.
HD	High-definition – HD video has higher resolution and quality than standard-definition.
Hough Transformation	– A feature extraction technique used image processing. It is proposed to find objects of certain shapes by a voting procedure.
IOU	Intersection over Union – an evaluation metric used to measure the accuracy of an object detector on a particular dataset.
KNN	K-nearest neighbors – A search strategy that aims at finding the point in a given set that is closest (or most similar) to a given point.
LIDAR	Light detection and ranging – A method for measuring distances (ranging) by illuminating the target with laser light and measuring the reflection with a sensor.
NMS	Non-maximum suppression – A post-processing method of object detection to remove duplicate detection.
RTSP	Real-time streaming protocol – A network control protocol designed for use in entertainment and communications systems to control streaming media servers.
SPAT	Signal phasing and timing
Softmax	A function that converts a vector of K real numbers into a probability distribution.