ANALYZING TURNING MOVEMENT COUNTS AT INTERSECTIONS THROUGH MULTI-CAMERA GROUND-PLANE REASONING

SAJJAD PAKDAMANSAVOJI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

YORK UNIVERSITY

TORONTO, ONTARIO

APRIL 2025

© SAJJAD PAKDAMANSAVOJI, 2025

Abstract

Classifying vehicle trajectories at intersections, known as turning movement counts (TMC), is a critical task for traffic management. Traditional approaches rely on a detect, track, count (DTC) paradigm that employs rule-based methods on image-plane data from a single camera. In this thesis, we propose a novel maximum likelihood approach that operates on the ground plane to perform trajectory classification. Our method demonstrates superior performance compared to image plane techniques and shows promising preliminary results for integrating multi-camera data on ground plane at the counting stage.

Acknowledgement

This thesis has greatly benefited from the support and assistance received during its development. I express my sincere gratitude to my supervisor, Professor James H. Elder, for his invaluable expertise in shaping the research questions and methodology. His insightful feedback was instrumental in enhancing the quality of my work, encouraging more precise thinking and elevating the overall standard of the research.

I am also deeply grateful to my colleagues, Kumar Vaibhav Jha and Maria Koshkina, for their intellectual support and collaborative spirit in our lab. The numerous discussions we shared and the joint projects we undertook have been instrumental in shaping my research. Their insights and contributions have undeniably played a crucial role in the evolution and success of my work.

Contents

Abstract ii

Acknowledgement iii

Contents iv

List of Tables vii

List of Figures ix

1 Introduction 1

1.1 Definition . . . 1

1.2 Imperative of TMC . . . 1

1.3 Research Motivation . . . 1

1.4 Background . . . 2

1.5 Performance of Previous Systems . . . 3

1.6 Our Approach . . . 3

2 Datasets 6

2.1 CityFlowV2 . . . 6

2.2 Trans-Plan Traffic Dataset . . . 7

2.3 Region of York Dataset . . . 8

2.4 Summary . . . 9

3 Geometric Calibration 10

3.1 Camera Model . . . 10

3.2 Camera Calibration . . . 10

3.3 Correcting Lens Distortion . . . 10

3.4 Region of Interest . . . 11

3.5 Back-Projection to Ground Plane . . . 12

3.6 Homography Graphical User Interface . . . 12

4 Object Detection 14

4.1 Background . . . 14

4.2 Methods . . . 14

4.2.1 Intersection over Union (IoU) . . . 14

4.2.2 YOLOX (You Only Look Once Extended) . . . 15

4.2.3 CenterNet (Keypoint Triplets for Object Detection) . . . 16

4.2.4 Cascade R-CNN (Cascaded Region Proposal Network) . . . 17
4.2.5 InternImage . . . 18
4.2.6 Comparison . . . 19
4.3 Evaluation . . . 19
4.4 Results . . . 20
4.4.1 Object Detection under Data Shift . . . 20
4.4.2 Detector Performance by Region: Intersection vs. Approaches . . . 22
4.5 Conclusion . . . 23

5 Multiple Object Tracking **24**
5.1 Background . . . 24
5.2 Method . . . 24
5.2.1 Causal vs Non-Causal Tracking . . . 24
5.2.2 Motion Model . . . 25
5.2.3 Appearance Model . . . 25
5.2.4 Association Model . . . 25
5.2.5 SORT . . . 25
5.2.6 Deep SORT . . . 26
5.2.7 ByteTrack . . . 26
5.3 Evaluation . . . 29
5.4 Results . . . 31
5.4.1 Evaluation of Detector-Tracker Combinations . . . 31
5.4.2 Tracking Performance by Region: Intersection vs. Approaches . . . 31
5.4.3 Tracking Performance by Weather Condition . . . 33
5.5 Conclusion . . . 34

6 Single-Camera Trajectory Classification **35**
6.1 Background . . . 35
6.2 Method . . . 35
6.2.1 Movements of Interest . . . 35
6.2.2 Vehicle Trajectory . . . 36
6.2.3 Trajectory Resampling . . . 37
6.2.4 Trajectory Filtering . . . 38
6.2.5 Contour Mapping Measure (CMM) . . . 39
6.2.6 Turning Movement Classification . . . 39
6.2.7 Single-Camera Prototype Selection . . . 40

6.2.8 Trajectory Classification . . . 42
6.2.9 KDE Bandwidth Optimization . . . 44
6.3 Evaluation . . . 50
6.3.1 Single-Camera TMC Evaluation . . . 50
6.3.2 Single-Camera TMC Bias . . . 50
6.4 Results . . . 50
6.4.1 Camera Plane vs Ground Plane Prototype Selection and Classification . . . 50
6.4.2 Impact of Camera Sway . . . 52
6.4.3 Impact of Fine-Tuning the Detector . . . 52
6.4.4 Impact of Weather and Illumination . . . 53
6.4.5 Failure Modes . . . 53
6.5 Conclusion . . . 59
6.5.1 Superiority of Ground Plane Reasoning for TMC . . . 59
6.5.2 Superiority of Maximum Likelihood Method . . . 60
6.5.3 Impact of Camera Sway . . . 60
6.5.4 Impact of Fine-Tuning the Detector . . . 60
6.5.5 Failure Modes . . . 60
6.5.6 Assumptions . . . 60

7 Multi-Camera Trajectory Classification 61
7.1 Background . . . 61
7.2 Method . . . 61
7.2.1 Multi-Camera Prototype Selection . . . 61
7.2.2 Multi-Camera Integration at the Counting Level . . . 62
7.3 Evaluation . . . 62
7.3.1 Multi-Camera TMC Evaluation . . . 62
7.3.2 Multi-Camera TMC Bias . . . 62
7.4 Results . . . 62
7.4.1 Integration at the Counting Level . . . 62
7.4.2 Evaluation of Multi-Camera system . . . 63
7.5 Conclusion . . . 64
7.5.1 Multi-Camera integration for TMC . . . 64

8 Future Work 65

List of Tables

1 Summary of the performance of previous approaches and their datasets. 4
2 Summary of installation details of the three datasets used. 9
3 Summary of sensor configuration of the three datasets used. 9
4 Summary of calibration details of the three datasets. 10
5 Average number of keypoints and resulting reprojection errors for estimated homographies on our three datasets. All orthophotos are 640 × 640 pixels. 13
6 Object size categories defined in MSCOCO. 20
7 Performance of pre-trained detectors on the Region of York dataset. 21
8 Course-grained performance of fine-tuned detectors on the Region of York dataset. 22
9 Fine-grained performance of fine-tuned detectors on the Region of York dataset. . 22
10 Performance of course-grained fine-tuned and pre-trained detectors within and outside the intersection boundaries on the validation set of the Region of York Dataset. . . 23
11 2D tracking results on validation set with pre-trained detectors on Region of York data. 31
12 2D tracking results on validation set with course-grained fine-tuned detectors on Region of York data. 31
13 Performance of course-grained pre-trained and fine-tuned detectors paired with trackers for tracks within and outside the intersection on the validation set of the Region of York Dataset. 33
14 Performance of fine-tuned InternImage and ByteTrack over the validation set of the Region of York dataset. 34
15 Assignment of MoI numbers based on entry and exit edges of ROI. 35
16 Optimal bandwidth for Gaussian kernel density estimation across different camera instances in the three datasets. 45
17 TMC MAE across four classification methods on both camera and ground-plane. We evaluate on the training partition of the CityFlow dataset and on the validation partitions of Trans-Plan and Region of York datasets. 51
18 Turning movement classification percent error of ground-plane counters across different sway conditions of the Trans-Plan dataset. 52
19 TMC performance (mean absolute error and mean bias over turning movement classes) of our ground-plane ML counter with pre-trained and fine-tuned systems on the Region of York validation partition. 53
20 TMC MAE of ground-plane ML on the validation partition of Region of York dataset under variable weather conditions. 53

22 (a) TMC error rates on Region of York training data by MoI and camera. (b) MoI to camera assignment. 63

23 TMC performance (mean absolute error and mean bias over turning movement classes) of single- and multi-camera (weak fusion) systems on the Region of York test partition. Oracle result comes from ground truth detection and tracking with our multi-camera system. 64

List of Figures

1 An overview of our TMC pipeline. 5

2 Camera image (upper row) and orthophoto (lower row) of four example intersections from the CityFlowV2 dataset. 7

3 Camera image (upper row) and orthophoto (lower row) of four example intersections from the Trans-Plan dataset. 8

4 Camera image (upper row) and orthophoto with overlaid reprojected camera image (lower row) from the 4 short-range cameras of the Region of York dataset. 9

5 Example of correcting lens distortion on a frame from the CityFlowV2 dataset. . . 11

6 Camera view (top) and aerial view (bottom) of four-sided polygon ROIs. 11

7 Homography GUI layout with scene from CityFlowV2. 12

8 Paired points selected from corresponding camera and orthophoto views. Annotated points are marked in blue, while reprojected points from the alternate perspective are in red. Example is from the CityFlowV2 dataset. 13

9 Example of predicted (red) and ground truth (green) bounding box of a vehicle. Intersection area is marked with blue. 15

10 Samples of bounding boxes detected by YOLOX. 16

11 Samples of bounding boxes detected by CenterNet. 17

12 Samples of bounding boxes detected by Cascade R-CNN. 18

13 Samples of bounding boxes detected by InternImage. 19

14 Course-grained performance of fine-tuned detectors on the validation set of the Region of York dataset as a function of training epoch. 21

15 Fine-grained performance of fine-tuned detectors on the validation set of the Region of York dataset as a function of training epoch. 22

16 Region of interest polygon for the Region of York dataset. 23

17 Example tracks generated by ByteTrack on the Region of York dataset. 28

18 Example tracks generated by SORT on the Region of York dataset. 28

19 Example tracks generated by Deep SORT on the Region of York dataset. 28

20 Vertices (v_i) and edges (e_i) of a 4 approach intersection and the 12 MoIs. 36

21 Contact point of vehicles marked in both camera view and ground-plane. 37

22 Resampled tracks in camera (a-b) and ground plane (c-d) domain. 38

23 Example of fragments of a track differentiated by color before (a) and post (b) filtering. The tracks were generated using the pre-trained InternImage detector and ByteTrack on the validation partition of the Region of York dataset. The highlighted quadrilateral indicates the RoI. 39

24 Prototype selection - Trans-Plan dataset. 41
25 Movement prototypes selected via our methodology and color marked by MoI. . . 42
26 Log probability of held-out validation data (Trans-Plan DBR segment) under a KDE model as a function of the bandwidth of the Gaussian kernel, in the image (a) and ground-plane (b) domain. 45
27 Class conditional density estimations for MoIs 1-6 for the ground-plane model on the Region of York dataset. Bright yellow regions signify areas of higher probability, while dark purple regions indicate areas of lower probability. 46
28 Class conditional density estimations for MoIs 7-12 for the ground-plane model on the Region of York dataset. Bright yellow regions signify areas of higher probability, while dark purple regions indicate areas of lower probability. 47
29 Class conditional log density estimations for MoIs 1-6 for the ground-plane model on the Region of York dataset. Bright yellow regions signify areas of higher probability, while dark purple regions indicate areas of lower probability. 48
30 Class conditional log density estimations for MoIs 7-12 for the ground-plane model on the Region of York dataset. Bright yellow regions signify areas of higher probability, while dark purple regions indicate areas of lower probability. 49
31 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 3 (c) instead of the correct value of 11 (b). 55
32 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 3 (c) instead of the correct value of 11 (b). 55
33 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 7 (c) instead of the correct value of 11 (b). 55
34 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 7 (c) instead of the correct value of 11 (b). 56

35 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 6 (c) instead of the correct value of 5 (b). 56
36 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 6 (c) instead of the correct value of 5 (b). 56
37 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 8 (c) instead of the correct value of 11 (b). 57
38 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 12 (c) instead of the correct value of 11 (b). 57
39 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 11 (b) while the track, due to re-id error, is a combination of MoI 2 (c) and 7 (d). . 58
40 An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 11 (b) while the track, due to re-id error, is a combination of MoI 5 (c) and 11 (d). 59
41 Prototype tracks captured from each of the four short-range cameras in the Region of York dataset over the same time interval. Each camera captures certain MoIs better than others. 61
42 Prototype tracks captured with individual and multiple cameras in the Region of York dataset over the entire training period. Note how selecting prototypes using all cameras results in a more complete set of prototypes. 61
43 MoIs assigned to each of the four short-range cameras in the Region of York dataset. 63

1 Introduction

TMC is a pivotal component of intelligent transportation systems (ITS). This thesis delves into this component, leveraging ground-plane reasoning.

1.1 Definition

TMC refer to the recording of the number of vehicles making specific turning movements at an intersection, including left turns, right turns, and straight movements. This provides a detailed view of traffic flow patterns at intersections, capturing the frequency and direction of vehicular movements. It is a fundamental aspect of traffic analysis used to assess and understand traffic dynamics.

1.2 Imperative of TMC

The need for advanced TMC analysis is multifaceted. In traffic management, precise TMC data is essential for optimizing signal timings, reducing congestion, and improving road safety. These data support the development of adaptive signal control systems that respond in real-time to changing traffic conditions. In urban planning, TMC data is used to design efficient road networks, plan pedestrian pathways, and strategize future urban expansions. TMC data help predict traffic demands and assess the impact of new infrastructure on traffic patterns.

1.3 Research Motivation

Recent advancements in the field of TMC have been driven by developments in core computer vision tasks, such as object detection and multiple object tracking (MOT). Traditional methods for detection, tracking, and classification primarily operate in the two-dimensional image domain, relying on image coordinates and often overlooking geo-spatial constraints. This leads to several limitations:

1. Kinematic information is distorted by perspective projection.
2. Kinematics are affected by varying camera parameters.
3. Understanding occlusions, which are inherently 3D, becomes complex.
4. Integrating data from sensors in different positions is challenging.

Geolocation information for objects can be obtained using intrinsically and extrinsically calibrated cameras, especially when objects move on known surfaces like flat roads or when a terrain model is available. Transitioning to ground-plane coordinates, instead of image coordinates, offers significant

potential. It makes room for the integration of detection and tracking data from multiple cameras within the same scene by projecting data onto a unified coordinate system, eliminating the limitations discussed above [87, 28, 26, 2].

1.4 Background

TMC has historically been conducted through manual means or via fixed sensors at intersections. Inductive loop detectors, which are reliable yet costly and susceptible to damage from environmental conditions such as freeze/thaw cycles, have been commonly employed [38]. Roadside radar, which offers broader trajectory data, are another option, though they provide limited details [88]. In terms of vehicle classification, these sensors do not offer precise categorization. Since the 1990s, traffic surveillance cameras have been adopted for real-time traffic forecasting and safety assessments, becoming more prevalent due to their cost-effectiveness and the amazing performace of advanced deep learning and computer vision techniques [48, 17, 59, 34]. Typically, vision-based TMC methods utilize a 3 stage detection, tracking, and counting (DTC) framework.

In DTC methods, trajectory classification is used to analyze vehicle movements. Classification methods in the literature can be grouped into line-based, zone-based, field map-based, and prototype matching approaches. In line-based methods [54, 60], trajectories are categorized by their intersections with predefined virtual lines. However, these methods require detailed annotation for each intersection and impose hard restrictions on the start or end of trajectories, making them prone to track fragmentation. Zone-based methods [23, 76] classify trajectories based on vehicle entries and exits from designated zones, with one variation incorporating a central zone to account for shorter trajectories [76]. Similar to line-based methods, zone-based approaches demand extensive annotation and enforce strict location constraints, but are more robust under fragmentation of tracks. Field map-based methods [37, 84] use manually drawn movement models to build 2D scoring functions on the camera view. These functions rank inference trajectories based on their proximity to the hand-drawn movements and their completeness. These approaches are limited by the arbitrary selection of scoring functions. Prototype matching approaches define representative trajectories either manually [25, 44], semi-automatically through line or zone intersections [43, 1], or via unsupervised clustering [43, 72]. New trajectories are matched to these prototypes using metrics such as cosine distance, Hausdorff distance [72], longest common subsequence (LCSS) [66], nearest neighbor techniques [1], or combinations of these methods [25, 44]. Since these approaches represent movements with only a few tracks, they fail to cover all possible variations, and their extraction depends on heuristic distance measures or clustering methods. A cascaded approach that integrates zone-based and prototype matching methods has also been proposed [64]. While

this method relaxes some of the strict assumptions of zone-based techniques, it still inherits the shortcomings of prototype matching.

An important distinction in DTC systems is the choice between single-camera and multi-camera configurations. Single-camera approaches are inherently limited in field of view and prone to occlusion, which can lead to missed vehicles or ambiguous tracking [73]. In contrast, multi-camera systems offer broader coverage and multiple viewpoints, mitigating blind spots and occlusions and thereby improving the reliability and accuracy [68]. Multi-camera methods can further be categorized by their perspective: some operate on each camera's image plane independently, while others transform all views into a common ground-plane bird's-eye view (BEV) representation. The ground-plane approach projects detections onto a unified overhead perspective of the intersection, reducing perspective distortion and simplifying cross-camera vehicle association; this strategy has been shown to significantly improve multi-object detection and tracking performance by alleviating occlusions and missed detections [71, 54] while its effect of TMC systems is yet to be determined. In contrast, image-plane methods must contend with differing camera angles and perspective warping, making data fusion and consistent tracking across views more challenging [73].

1.5 Performance of Previous Systems

Previous systems generally used proprietary datasets from a single camera and reported inconsistent performance metrics, complicating direct comparison. Several studies relied on datasets from the AI City Challenge [55], reporting a combined metric of efficiency and effectiveness that is not directly comparable to TMC error. Other methods reported counting errors in the range of 4% to 16% but did not share their datasets publicly. Moreover, these studies did not include metrics indicating the degree of overcounting or undercounting. Table 1 includes a summary of the performance of previous works and their corresponding datasets.

1.6 Our Approach

In our research, we adopt the conventional DTC framework (Figure 1), using a cutting-edge deconvolution-based foundation model detector, InternImage [75], and ByteTrack [86], a tracker based on Kalman filter principles. Our counting methodology is a hybrid technique that first identifies prototype tracks using an intersection polygon and then employs an unsupervised probabilistic method to estimate prototype densities, effectively handling shorter query tracks. Density estimation is performed on the ground-plane to mitigate perspective projection errors and to integrate data from multiple cameras. Finally, a weak multi-camera fusion approach is applied to classify trajectories based on the best available vantage point.

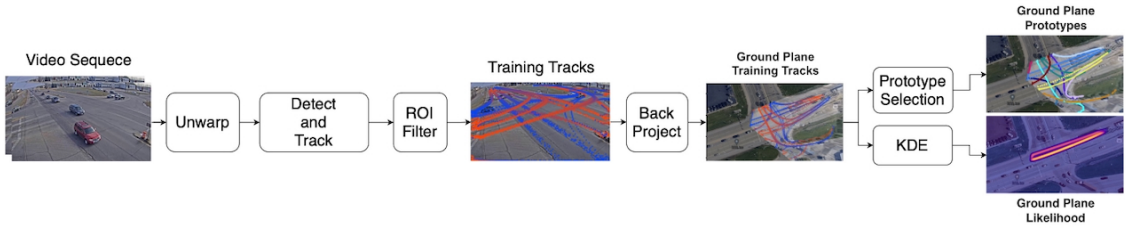
Method	Category	Dataset	Metric	Value
[60]	Line-based	CityFlow [70]	Weighted combination of system efficiency and accuracy [55]	0.93
[25]	Line-based	AI City Challenge - Track 1 [56]	Weighted combination of system efficiency and accuracy [55]	0.93
[37]	Field-Map	AI City Challenge - Track 1 [56]	Weighted combination of system efficiency and accuracy [55]	0.84
[84]	Field-Map	AI City Challenge - Track 1 [56]	Weighted combination of system efficiency and accuracy [55]	0.92
[25]	Prototype Matching	AI City Challenge - Track 1 [56]	Weighted combination of system efficiency and accuracy [55]	0.94
[44]	Prototype Matching	AI City Challenge - Track 1 [56]	Weighted combination of system efficiency and accuracy [55]	0.94
[44]	Zone-based	AI City Challenge - Track 1 [56]	Weighted combination of system efficiency and accuracy [55]	0.81
[72]	Zone-based	AI City Challenge - Track 1 [56]	Weighted combination of system efficiency and accuracy [55]	0.92
[64]	Prototype matching	Proprietary dataset	Counting error	16.1%
[76]	Zone-Based	Proprietary dataset	Counting error	3.9%

Table 1: Summary of the performance of previous approaches and their datasets.

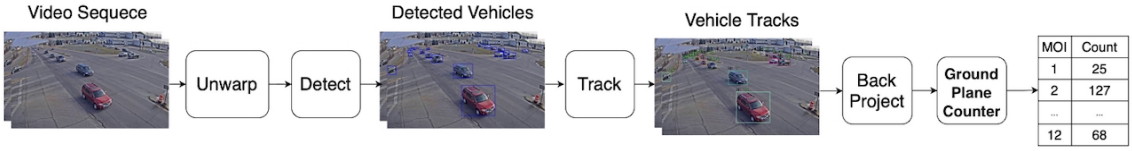
Our approach addresses several shortcomings of previous works. It requires only a lightweight annotation to define the intersection polygon, unlike line-based, zone-based, and field-map-based methods that necessitate detailed annotations for each intersection. Moreover, the use of a probabilistic density function eliminates the need for heuristic distance metrics and scoring functions in trajectory comparison and prototype extraction, allowing movement to be represented as a density estimated over many tracks rather than a few prototypes. Additionally, the multi-camera mode enables the system to select the optimal vantage point for trajectory classification, reducing failures caused by occlusion.

Our contribution lies in orchestrating a system that brings together multi-camera configuration, ground-plane perspective, and unsupervised probabilistic models to address the TMC problem. While each of these approaches has been applied individually within the DTC framework to various extents, our system is the first to fully integrate them all simultaneously.

Our TMC pipeline involves several steps: correcting lens distortion, defining regions of interest, solving the camera-to-ground mapping, detecting and tracking objects in the image plane, mapping image trajectories to the ground-plane, extracting movement prototypes, classifying trajectories, and evaluating performance with relevant metrics. The subsequent sections will detail each component of this pipeline.



(a) Training pipeline



(b) Inference pipeline

Figure 1: An overview of our TMC pipeline.

2 Datasets

Most existing traffic intersection datasets (e.g., [78, 56, 55, 57, 83]) offer only one camera view per intersection with the exception of the multimodal A9-Intersection [16] and the HoloVIC [46] datasets which provide multiple views per intersection, though the field of view of their cameras is narrow and insufficient for comprehensive traffic monitoring. In this study, we evaluate our TMC methods using the CityFlow V2 [57] dataset, a typical representative of these existing datasets.

We also introduce two new datasets to address limitations in current datasets. Existing datasets often have cameras mounted high in the infrastructure, which minimizes occlusions [54]. However, typical traffic surveys use cameras mounted temporarily at lower heights (under 5 meters). To address this, we created the Trans-Plan dataset with a single-view, low-vantage, wide-angle setup, in collaboration with our industry partner Trans-Plan. Our Region of York dataset provides four synchronized views of the same intersection to examine the impact of multiple camera views on TMC accuracy.

Ground truth turning movement counts were manually collected for all three datasets. Orthophoto imagery from Google Maps was used to establish point correspondences and determine homographies mapping camera planes to planar approximations of the ground terrain.

We divided the Trans-Plan and Region of York datasets into training and validation partitions. Although we do not use supervised training methods, which reduces the risk of overfitting, we do employ unsupervised optimization methods prior to deployment. Validation data are thus crucial for estimating performance at deployment. Due to the limited video duration available for each intersection in the CityFlow V2 dataset, we chose not to partition it, which could lead to optimistic results, since in real deployment the statistics might drift after unsupervised training. Thus the results reported for CityFlow V2 should be seen as an upper bound on performance, that could be approached by employing an optimized online retraining algorithm that handles any drift in the statistics of the data. Detailed information on each dataset is provided below.

2.1 CityFlowV2

The AIC22 Benchmark Dataset (CityFlowV2) is a collection of traffic surveillance data, acquired from a network of 46 cameras installed in real-world environments, capturing the dynamics of 880 annotated vehicles across six different scenarios. This dataset is notable for its comprehensive coverage, offering 215.03 minutes of video data segmented into training (58.43 minutes), validation

(136.60 minutes), and test (20.00 minutes) scenarios. The dataset includes detailed geo-location data, with GPS coordinates for approximate camera locations, enhancing the dataset's utility for spatial analysis. The resolution of each video is at least 960p and the majority of the videos have a frame rate of 10 FPS. In out study, a subgroup of eight videos has been selected, specifically from cameras 5, 35, 41, 42, 43, 44, 45, and 46, totaling 23m30s of footage. These cameras have a wide field of view [57, 70]. The required annotations for turning direction classification were added as part of our research. A few example scenes from this dataset are shown in Figure 2.



(a) C06S044 (b) C06S043 (c) C05S042 (d) S06C041

Figure 2: Camera image (upper row) and orthophoto (lower row) of four example intersections from the CityFlowV2 dataset.

2.2 Trans-Plan Traffic Dataset

This proprietary dataset (Fig 3), consisting of motor vehicle movements at suburban intersections, was compiled by Trans-Plan, an industry collaborator, at four intersections in the Great Toronto Area. These intersections are located at Dundas Street and 9th Line (D9L), Dundas Street and Winston Park Drive (DWP), Eglinton Avenue West and Creditview Road (ECR), and Dundas Street and Bronte Road (DBR). The dataset is composed of 60-minute video recordings, captured at 4K resolution and 60 frames per second by a stationary camera temporarily set up at each location at approximate height of 4.5 meters. The recordings are divided into two 30 minute partitions; the first for training and the second for validation. I used an orthophoto from Google Maps to determine a homography mapping for each intersection. The videos have been annotated with turning movement counts by Trans-Plan. Labels were drawn from 12 movement categories; 4 initial direction of movement (N, S, E, W) x 3 turning classes (left, straight, right).



(a) D9L (b) DWP (c) DBR (d) ECR

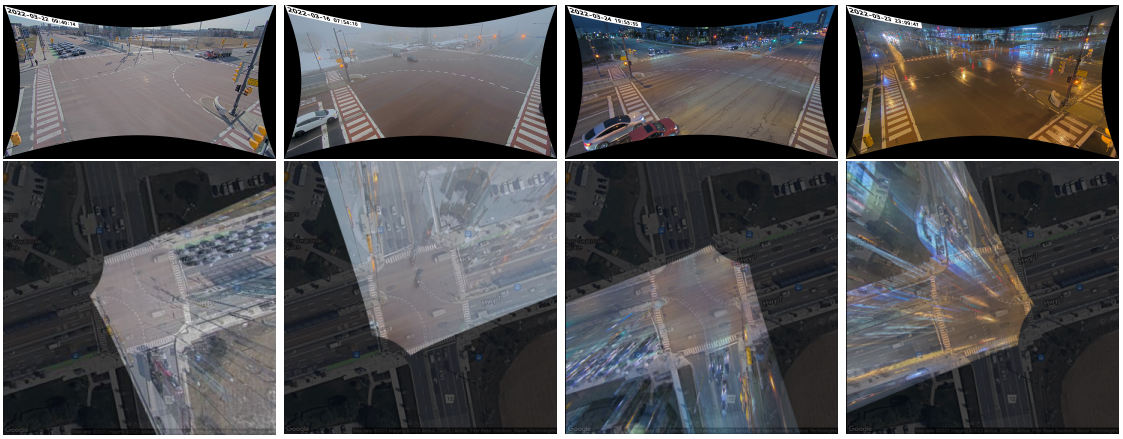
Figure 3: Camera image (upper row) and orthophoto (lower row) of four example intersections from the Trans-Plan dataset.

2.3 Region of York Dataset

The third dataset we used was created from a collaborative project involving York University, the University of Toronto, and the Region of York. This dataset was collected at the intersection of Highway 7 and Leslie Street and was designed to support research on multi-camera traffic analytics.

The setup included four short-range 4K 60fps cameras installed at the four corners of the intersection, two long-range cameras focused on traffic approaching from the west, and two LiDAR systems positioned to provide 3D data coverage of both the intersection and the western approach. This setup ensured that every location within the intersection was within the field of view of multiple cameras.

This dataset consists of 32 video segments, divided into 25 training segments and 7 validation segments, capturing a broad spectrum of weather conditions (ranging from dry to snow) and illumination scenarios (including sunny, cloudy, nighttime, dusk, and dawn), to represent the challenging conditions often encountered in real-world environments. The intersection's layout, featuring eight lanes running east-west (including two bus lanes and dedicated turn lanes) and six lanes running north-south (with dedicated turn lanes), along with dedicated bicycle lanes and a complex pedestrian crosswalk system, provides a good example of a modern city traffic scenario. Ground truth annotations for both 2D and 3D detection, tracking, and counting have been added as part of this study. Example images from this dataset are shown in Figure 4.



(a) Camera 1 (b) Camera 2 (c) Camera 3 (d) Camera 4

Figure 4: Camera image (upper row) and orthophoto with overlaid reprojected camera image (lower row) from the 4 short-range cameras of the Region of York dataset.

2.4 Summary

The three datasets selected for this study provide a good platform for evaluating the effectiveness of image-plane versus ground-plane reasoning in TMC. The datasets differ significantly in camera installation height, illumination conditions, weather scenarios, resolution, FoV, and geographical location, as summarized in Table 2 and 3. The Region of York dataset's multi-camera setup allows for the exploration of ground-plane reasoning with multiple viewpoints, while CityFlow and TransPlan are suitable for examining the benefits of ground-plane reasoning in single viewpoint scenarios.

Dataset	**Installation Height**	**Illumination**	**# Cameras**	**LiDAR**
CityFlowV2	Unknown	Clear/Sunny	1	x
Trans-Plan	$\sim$ 4.5 m	Clear/Sunny	1	x
Region of York	$\sim$ 10 m	Diverse	6	✓

Table 2: Summary of installation details of the three datasets used.

Dataset	**Resolution**	**FPS**	**Horizontal FoV**	**Vertical FoV**
CityFlowV2	>960p	10	Unknown	Unknown
Trans-Plan	4K	60	122°	94°
Region of York	4K	60	90°	49°

Table 3: Summary of sensor configuration of the three datasets used.

3 Geometric Calibration

3.1 Camera Model

The camera model establishes a relationship between 3D world coordinates and their corresponding 2D image coordinates. The camera maps a point's world coordinates (X, Y, Z) to pixel coordinates (x_p, y_p) in the camera image. This depends on the camera's position and orientation in the world frame, defined by translation and rotation parameters (extrinsic parameters, R and t). These parameters transform the point's coordinates to the camera coordinates (X', Y', Z'), where the camera frame is a 3D system with its optical center at the origin, and optical axis along the positive Z-axis. This process is represented in Equation 1.

$$\lambda \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = K \, [R|t] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K \begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} \tag{1}$$

3.2 Camera Calibration

Calibration was performed for the Region of York and Trans-Plan datasets. Camera intrinsics were calibrated in the lab for all 5 cameras, using the MATLAB calibration toolbox [31] and a checkerboard pattern. Due to the wide FOV of the cameras, we employed a fisheye camera model. Intrinsic parameters of the CityFlow cameras are provided as part of the dataset. We do not attempt a full estimation of extrinsic parameters but instead estimate for each site and each camera a homography mapping from the camera plane to a planar approximation of the ground plane (see Table 4).

Dataset	**Camera Make**	**Camera Model**	**Method**
CityFLowv2	Unkown	Unknown	Unknown
Trans-Plan	GoPro	Hero8	MATLAB
Region of York	Axis	Q1798LE	MATLAB

Table 4: Summary of calibration details of the three datasets.

3.3 Correcting Lens Distortion

Surveillance cameras typically have a wide field of view, resulting in barrel distortion in video footage. This distortion may affect the accuracy of detection and tracking algorithms, which usually assume no lens distortion. Therefore, we use video preprocessing techniques to correct barrel distortion, using a fisheye lens model. We used the OpenCV python library [11] to undistort

the videos from Trans-Plan and CityFlowV2. We used MATLAB [31] to undistort videos from the Region of York dataset. An example of lens distortion correction is shown in Figure 5. The distortion at the frame corners (Figure 5 (b)) is likely due to errors in the intrinsic parameters provided in the dataset or from using a fisheye model, which is better suited for cameras with a wide FoV. Since the CityFlowV2 dataset does not provide FoV or calibration details, this is the best we can achieve for undistorting the frames.

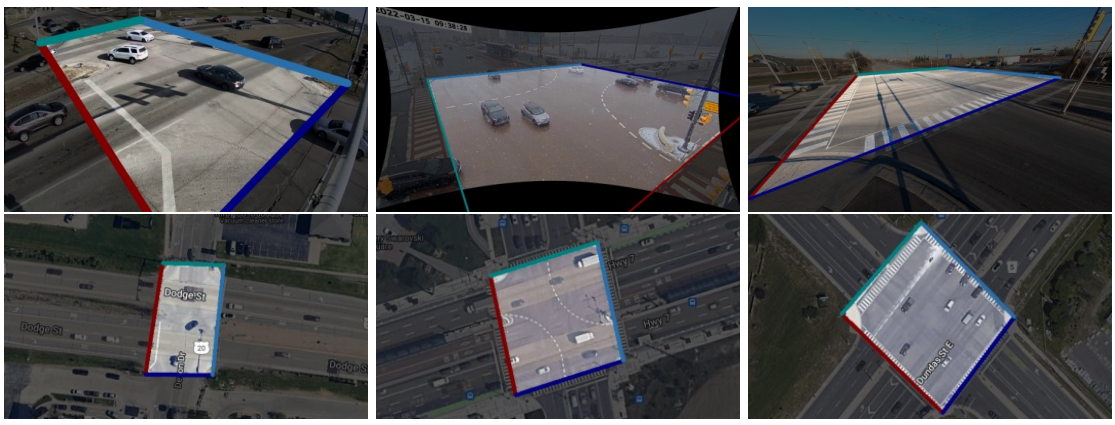


(a) Original (b) Corrected

Figure 5: Example of correcting lens distortion on a frame from the CityFlowV2 dataset.

3.4 Region of Interest

The selection of a Region of Interest (ROI) is essential for efficient object detection, tracking, and TMC. It directs algorithms to areas of traffic flow, excluding irrelevant regions. We define the ROI as a four-sided polygon. Vertices are positioned at the corners of the intersection. The edges of the polygon align with the crosswalks in each direction and are perpendicular to the approaching traffic flow. Examples of selected ROIs are shown in Figure 6.



(a) CityFlowV2 S06C0041 (b) Region of York S04Sc1 (c) Trans-Plan D9L

Figure 6: Camera view (top) and aerial view (bottom) of four-sided polygon ROIs.

3.5 Back-Projection to Ground Plane

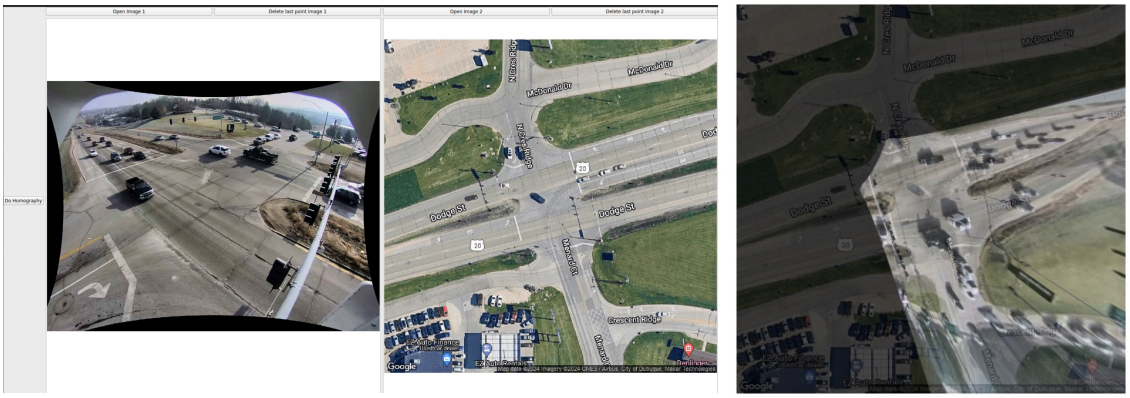
Road users maintain contact with the ground, which we approximate as planar. We use a homography H to back-project each point (x, y) from the camera view to a point (u, v) on the ground-plane.

$$c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad where \quad H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{2}$$

The homography matrix, H, is determined up to a scale factor, denoted as c in equation 2. Therefore, although H contains 9 elements (as a 3x3 matrix), it possesses 8 degrees of freedom meaning that at least 4 pairs of corresponding points in the camera view and orthophoto are required to estimate the homography. If more than 4 are available, a least squares solution is estimated [52].

3.6 Homography Graphical User Interface

We created a graphical user interface (Figure 7 (a)) in order to manually identify corresponding points in camera view and orthophoto points. The interface features five buttons: two for uploading camera and aerial view images, two for annotating point pairs, and one for initiating the algorithm that computes the homography estimation. The estimation algorithm is a least squares method from the OpenCV python library [32], cv2.findhomography(). Users initiate the process by uploading street and aerial view images. They then select corresponding points on both images, using features like crosswalks, bus stops, or lane markings. Once the homography estimation is completed, the application generates an image showing the camera image reprojected to and overlaid on the orthophoto (Figure 7 (b)). The interface is developed in Python using OpenGL [80] and QT5 [21]. Figure 8 shows the qualitative error of the reprojection method. Table 5 shows the reprojection error for each camera in our datasets.



(a) Homography GUI view

(b) Resulting overlaid image

Figure 7: Homography GUI layout with scene from CityFlowV2.



(a) Camera view

(b) Orthophoto

Figure 8: Paired points selected from corresponding camera and orthophoto views. Annotated points are marked in blue, while reprojected points from the alternate perspective are in red. Example is from the CityFlowV2 dataset.

Dataset	**Resolution** (pixels)	**Camera**	**# Points**	**Reprojection Error** **Camera View** (pixels)	**Orthophoto** (pixels)
TransPlan	3596 x 2158	D9L	25	13.2	3.5
		DBR	22	9.9	3.4
		DWP	16	18.3	3.8
		ECR	26	31.3	6.3
		Avg.	**22**	**18.2**	**4.3**
CityFlow	1280 × 960	C005	15	9.9	2.7
		C035	14	24.0	2.6
		C041	12	15.2	4.3
		C042	22	14.0	3.8
		C043	12	18.5	3.0
		C044	12	9.9	2.9
		C045	10	8.2	2.6
		C046	10	33.6	11.8
		Avg.	**13**	**16.7**	**4.2**
Region of York	4840 × 2726	Sc1	16	21.8	2.9
		SC2	20	16.3	2.8
		SC3	20	14.5	2.8
		SC4	16	10.6	2.8
		Avg.	**18**	**15.8**	**2.8**

Table 5: Average number of keypoints and resulting reprojection errors for estimated homographies on our three datasets. All orthophotos are 640 × 640 pixels.

4 Object Detection

Object detection involves predicting a set of bounding boxes $B = \{b_1, b_2, \ldots, b_N\}$ and corresponding class labels $L = \{l_1, l_2, \ldots, l_N\}$ for an image I. Each bounding box b_i is defined by its location and dimensions (x_i, y_i, w_i, h_i), where (x_i, y_i) are the coordinates of the top-left corner, and w_i and h_i denote the width and height, respectively. The class label l_i categorizes the object within b_i. We evaluate four deep learning object detection models: YOLOX [22], CenterNet [19], CascadeRCNN [62], and InternImage [75]. These models are trained on MSCOCO [42], a large-scale dataset widely used for tasks such as object detection, segmentation, keypoint detection, and captioning that supports 80 object classes. We narrow our analysis to 6 object classes: pedestrian, car, truck, bus, bicycle, and motorcycle.

4.1 Background

Early object detection methods relied on background subtraction, clustering [5, 6], Gaussian Mixture Models (GMM) [65, 81], or Mixture of Deformable Part Models [20]. Recent studies have shifted towards employing Convolutional Neural Networks (CNN), which are generally divided into one-stage and two-stage networks. Examples of the former include CenterNet [37], various YOLO implementations [25, 44, 72, 43, 66, 1, 12], RetinaNet [23], and NAS-FPN [77], which are noted for their fast inference speeds. In contrast, two-stage networks like Mask-RCNN [14] and Faster RCNN [43] often utilize feature aggregation methods such as Feature Pyramid Networks (FPN) to enhance accuracy through improved feature extraction. Recent innovations seek to boost detection accuracy and processing speed by introducing novel approaches within these architectures. These include detection augmentation strategies to reduce false negatives [43], and frame cropping based on previous bounding box locations to quicken inference [23, 51].

4.2 Methods

4.2.1 Intersection over Union (IoU)

IoU is a metric used to measure the accuracy of a predicted bounding box $b_{pred} = \{x_p, y_p, w_p, h_p\}$ against a ground truth bounding box $b_{gt} = \{x_g, y_g, w_g, h_g\}$ (Figure 9). It calculates the ratio of the area of overlap between the predicted bounding box and the ground truth bounding box to the area encompassed by both bounding boxes combined. Equations 3-6 collectively represent the steps required to compute the IoU metric.

$$x_{\text{inter-left}} = \max(x_p, x_g) \qquad y_{\text{inter-top}} = \max(y_p, y_g) \tag{3}$$

$$x_{\text{inter-right}} = \min(x_p + w_p, x_g + w_g) \qquad y_{\text{inter-bottom}} = \min(y_p + h_p, y_g + h_g) \tag{4}$$

$$w_{\text{inter}} = \max(0, x_{\text{inter-right}} - x_{\text{inter-left}}) \qquad h_{\text{inter}} = \max(0, y_{\text{inter-bottom}} - y_{\text{inter-top}}) \tag{5}$$

$$\text{IoU} = \frac{w_{\text{inter}} \times h_{\text{inter}}}{w_p \times h_p + w_g \times h_g - w_{\text{inter}} \times h_{\text{inter}}} \tag{6}$$

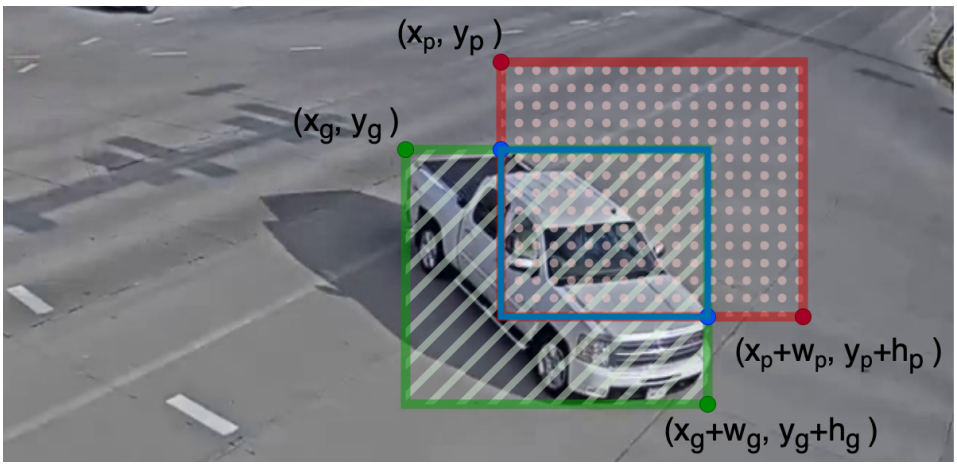
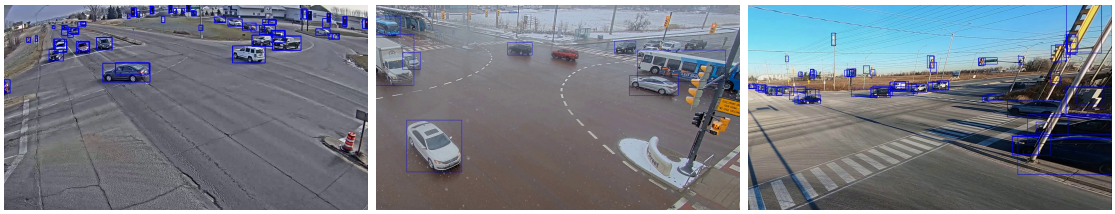


Figure 9: Example of predicted (red) and ground truth (green) bounding box of a vehicle. Intersection area is marked with blue.

4.2.2 YOLOX (You Only Look Once Extended)

YOLOX [22] adopts an anchor-free detection approach, reducing computational overhead needed for carefully selecting the anchor box sizes and aspect ratios. It simplified the network structure, compared with previous YOLO versions by using standard, well-optimized architectures that are commonly used in other state-of-the-art computer vision models such as CSPDarknet53 [10]. A key innovation is the decoupled head for object classification and bounding box prediction which enhances both accuracy and inference speed. Additionally, YOLOX incorporates the SimOTA (Simplified Optimal Transport Assignment) label assignment strategy. It is a technique used for assigning ground truth boxes to predicted boxes during training. SimOTA simplifies the process of matching predictions with ground truth by focusing on the optimization of the overall matching quality instead of following fixed intersection over union (IoU) thresholds or complex matching rules.

YOLOX's architecture typically uses as a backbone a variant of networks such as CSPDarknet [10], MobileNet [29], or ShuffleNet [85]. The neck aggregates features from different scales, often using feature pyramid networks (FPN). Its training employs aggressive data augmentation strategies like Mosaic and MixUp, and the SGD optimizer with a cosine learning rate schedule. An example of the objects detected by this model is shown in Figure 10.



(a) CityFlowV2 S06C045 (b) Region of York Seg01Sc1 (c) Trans-Plan D9L

Figure 10: Samples of bounding boxes detected by YOLOX.

4.2.3 CenterNet (Keypoint Triplets for Object Detection)

Traditional object detection models often predict the coordinates of the corners of a bounding box. CenterNet [19], however, focuses on predicting the exact center of the object. This center is defined as the point that lies in the middle of the object. During training, a heatmap ground truth is generated by placing Gaussians at the center of each object, with the peak aligning with the object's center. CenterNet is trained to predict these heatmaps for each class. During inference, the model outputs these heatmaps, where the peaks indicate the predicted centers of objects. The model then uses these centers to predict the corresponding bounding boxes and object classes. Once the center is detected, the model predicts the size (height and width) and the offset of the bounding box relative to the center point.

CenterNet's architecture is based on a convolutional neural network backbone, such as ResNet [27] or Hourglass [58], with minor modifications optimized for feature extraction. Standard convolutional operations typically aggregate information from local neighborhoods (e.g., max pooling). In contrast, CenterNet pools features not just from a local neighborhood, but from the entire horizontal and vertical extent of the object. This is done by applying separate pooling operations along the horizontal and vertical axes and combining the results.

Training involves a multi-part loss function: focal loss [41] for heatmap prediction, regression loss for size and offset prediction, and pull-push loss [35] for grouping keypoints of the same object. The pull component forces keypoints from the same object to be close in the feature space while the push component forces distance between keypoints from different objects. Data augmentation techniques like random cropping, scaling, and color jittering were used to enhance generalization. Examples of detected bounding boxes are shown in Figure 11. CenterNet performs poorly in Figure 11 (c), where a traffic light pole occludes the vehicle. This likely happens because the occlusion covers the bounding box center, removing key cues CenterNet relies on to predict the full bounding box.



(a) CityFlowV2 S06C045 (b) Region of York Seg01Sc1 (c) Trans-Plan D9L

Figure 11: Samples of bounding boxes detected by CenterNet.

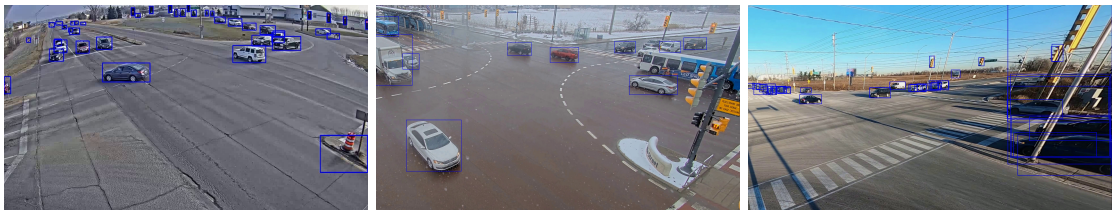
4.2.4 Cascade R-CNN (Cascaded Region Proposal Network)

Cascade R-CNN [62] enhances bounding box localization through a multi-stage detection architecture. This framework employs a cascading mechanism, where each stage refines the bounding box proposals from the previous stage. Each detector(stage) is trained at progressively higher IoU thresholds, aligning the region proposals with the ground truth more accurately.

The architecture includes several stages, each with a region proposal network (RPN) and a Region of Interest (RoI) head. The initial stage uses the RPN to generate preliminary proposals, which are then refined by the RoI heads in subsequent stages. Unlike traditional detectors with fixed IoU thresholds, Cascade R-CNN increases IoU thresholds at each stage, enhancing the precision of the final detections.

The training process is sequential, with each stage trained in sequence, using the output of the previous stage as input. This ensures focused training aligned with increasing IoU thresholds. To address the challenge of fewer positive samples at higher IoU levels, a resampling strategy is employed, maintaining an adequate number of positive samples for each stage. Each stage is trained using a multi-task loss combining classification loss and bounding box regression loss, enabling the model to learn object categories and precise localization simultaneously.

Cascade R-CNN significantly advances object detection with its precise localization, robustness to varying IoU thresholds, and adaptability to complex scenes. It excels in handling intricate object arrangements and occlusions by progressively refining proposals. Its innovative approach, robust training methodology, and precise localization establish Cascade R-CNN as a leading model in object detection, suitable for various applications. An example of bounding boxes detected by this model is shown in Figure 12.



(a) CityFlowV2 S06C045 (b) Region of York Seg01Sc1 (c) Trans-Plan D9L

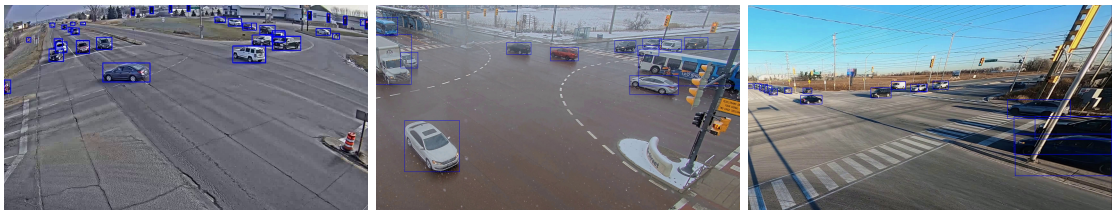
Figure 12: Samples of bounding boxes detected by Cascade R-CNN.

4.2.5 InternImage

InternImage [75] is a neural network model leveraging deformable convolutions (DCNv3). The DCNv3 operator dynamically adjusts its receptive field based on the input and task, reducing the rigid inductive biases of traditional CNNs. One of its unique improvements over traditional CNNs is the offset prediction mechanism. It predicts the positions where convolutional filters sample the input feature map. Another important improvement in DCNv3 is the introduction of a modulation mechanism. This mechanism controls the importance or weight of each pixel, effectively allowing the network to focus on the most relevant parts of the image.

InternImage's architectural design draws from advanced transformer components [74]. The model employs layer normalization [3] and feed-forward networks in its basic blocks and integrate them with the DCNv3 operator. The stem and downsampling layers adhere to conventional CNN designs, ensuring the feature representation and compatibility with different input scales. Similar to transformers, number of blocks stacked, channels, width and depth of this model is adjustable and selected by careful empirical analysis. DCNv3's architecture has been streamlined to reduce unnecessary complexity, which helps improve gradient flow. By optimizing the number of parameters and operations, DCNv3 reduces the chances of introducing bottlenecks or inefficiencies that can impede the flow of gradients through the network.

InternImage pre-training involves training the model on a vast dataset to capture general features and patterns that are useful across various tasks. This extensive pre-training allows InternImage to serve as a foundation model, providing a robust and versatile base that can be fine-tuned for specific applications with less data and effort. An example of its performance is shown in Figure 13.



(a) CityFlowV2 S06C045 (b) Region of York Seg01Sc1 (c) Trans-Plan D9L

Figure 13: Samples of bounding boxes detected by InternImage.

4.2.6 Comparison

YOLOX, CenterNet, Cascade R-CNN, and InternImage represent distinct paradigms within the object detection landscape, each with unique technical attributes. YOLOX, part of the YOLO family, is renowned for its real-time processing capabilities, utilizing an anchor-free mechanism to predict bounding boxes and class probabilities directly from feature maps. CenterNet adopts a minimalist approach, eliminating anchors and representing objects as points, specifically their center points, to simplify the detection pipeline. Cascade R-CNN features a multi-stage architecture that refines object proposals sequentially, using increasingly strict intersection-over-union (IoU) thresholds to enhance bounding box precision, albeit at a higher computational cost. InternImage diverges from traditional convolutional architectures by embedding deformable convolutions within its layers. This allows the model to dynamically adjust its receptive field based on the input, reducing inductive bias and enabling it to learn more generalized representations. InternImage scales effectively with increased parameters and training data, similar to large-scale vision transformers (ViTs), achieving state-of-the-art performance on benchmarks like COCO and ADE20K. Each model's design choices balance computational efficiency, accuracy, and the ability to handle varying scales and complexities inherent in object detection tasks.

4.3 Evaluation

The detection component is assessed based on its ability to correctly identify objects within frames. Key metrics include Precision, Recall, Average Precision, and mean Average Precision [4].

- **Precision:** Proportion of correctly identified positive instances among all instances identified as positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

- **Recall:** Proportion of actual positives correctly identified.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

- **Average Precision (AP):** Evaluates the precision-recall tradeoff for a single class by calculating the area under the precision-recall curve. A precision-recall curve plots the value of precision against recall for different confidence threshold values.

$$\text{AP} = \int_0^1 p(r)\, dr \tag{9}$$

where r = recall and $p(r)$ is the precision corresponding to recall r from the curve.

- **Mean Average Precision (mAP):** Mean of the AP scores for all classes, providing an overall effectiveness of detection across all categories.

$$\text{mAP} = \frac{1}{N} \sum_{n=1}^{N} AP_n \tag{10}$$

where AP = Average Precision and N = number of classes. It is common to report mAP for certain IoU thresholds as well where only detections with an IoU grater than a certain value are considered in the metric calculation. For example mAP@50 means mAP of detections with IoU equal or greater than 50%. We report mAP@50 and mAP@75 in our results sections. We also report mAP-size variants where the mAP is calculated only for boxes with area between certain thresholds (Table 6). For example mAP-Small is mAP calculated over detections with area less than 32^2, mAP-Large is calculated over boxes with area grater than 96^2 and mAP-Medium is calculated over the rest of the boxes with are between 32^2 and 96^2.

	Area of bounding box (Pixels^2)	
	Min	Max
Small	0	32^2
Med	32^2	96^2
Large	96^2	∞

Table 6: Object size categories defined in MSCOCO.

4.4 Results

4.4.1 Object Detection under Data Shift

We examine the performance of selected detectors, originally trained on the MS-COCO dataset [42], on unseen data. This analysis is essential because, despite achieving high accuracy on public benchmarks, these detectors often suffer performance declines under data shift. We evaluate four detectors: YOLOX, Cascade RCNN, CenterNet, and InternImage on the Region of York dataset as it has the necessary annotations. We average results across all validation segments and cameras, both short-range and long-range, and report mAP based on bounding box size: small (mAP-s),

medium (mAP-m), and large (mAP-l); following the size categories defined by the area of the bounding box as suggested by MS-COCO (Table 6). Our evaluation is presented in Table 7. Out of the 80 object classes supported by MSCOCO we evaluate on 6 (person, bicycles, motorcycle, car, bus, truck) that align with the taxonomy used in the Region of York dataset. Our findings indicate generally low off-the-shelf performance, especially for objects in the far field that appear small in the image (mAP-s), where the mAP drops to 0.001. InternImage performs best across all metrics.

Model	**mAP**	**mAP@50**	**mAP@75**	**mAP-s**	**mAP-m**	**mAP-l**
YOLOX	0.138	0.243	0.134	0.001	0.031	0.251
Cascade RCNN	0.153	0.263	0.154	0.001	0.038	0.266
CenterNet	0.132	0.253	0.129	0.001	0.036	0.232
InternImage	0.214	0.346	0.223	0.001	0.075	0.345

Table 7: Performance of pre-trained detectors on the Region of York dataset.

To improve model performance, we fine-tuned weights in the classification and regression heads of the detectors using the MMDetection repository [15]. These models, originally trained on the 80-class MS-COCO dataset, were first modified to align with our 6-class (course-grained) and 16-class (fine-grained) taxonomy by replacing the final Softmax layer of the classification head. We fine-tuned for 50 epochs, maintaining the original batch size and using a learning rate schedule with cosine annealing. The number of epochs was limited by our computational and time constraints, but extending the training could help explore better local optima for the models. This procedure significantly enhances performance across all object scales, highlighting the data shift between the MS-COCO and Region of York datasets. Figure 14 and 15 show how performance on validation data improves with each epoch of training and Table 8 and 9 show the best perforamnce achieved.

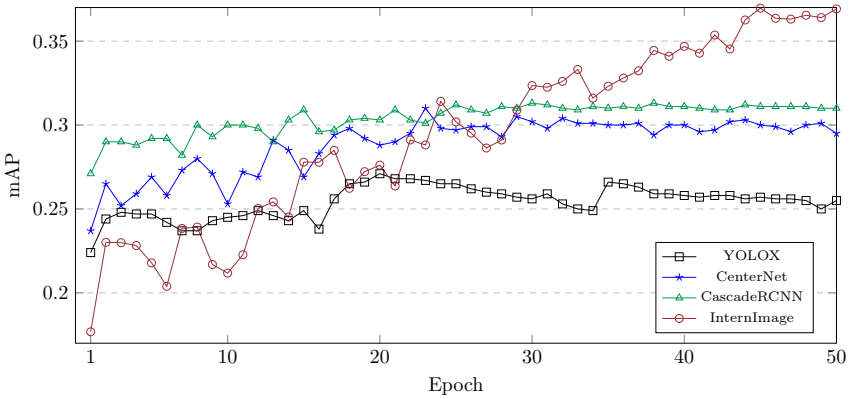


Figure 14: Course-grained performance of fine-tuned detectors on the validation set of the Region of York dataset as a function of training epoch.

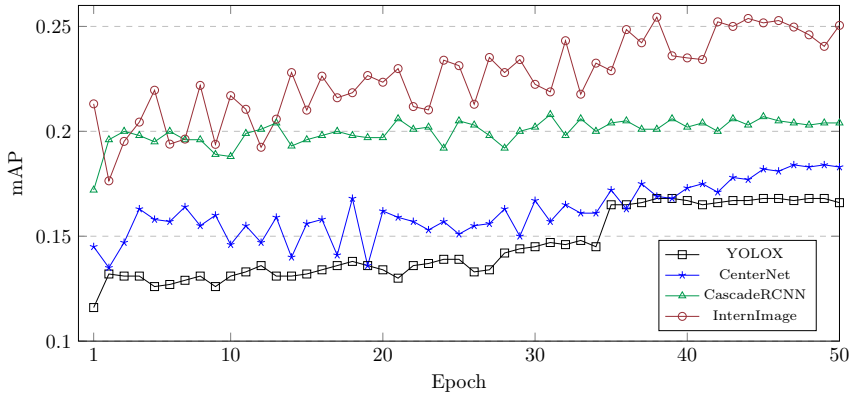


Figure 15: Fine-grained performance of fine-tuned detectors on the validation set of the Region of York dataset as a function of training epoch.

Model	mAP	mAP@50	mAP@75	mAP-s	mAP-m	mAP-l
YOLOX	0.271	0.444	0.277	0.016	0.100	0.468
Cascade RCNN	0.313	0.461	0.344	0.001	0.149	0.509
CenterNet	0.310	0.514	0.314	0.025	0.165	0.478
InternImage	0.369	0.511	0.417	0.002	0.205	0.566

Table 8: Course-grained performance of fine-tuned detectors on the Region of York dataset.

Model	mAP	mAP@50	mAP@75	mAP-s	mAP-m	mAP-l
YOLOX	0.168	0.244	0.188	0.003	0.063	0.319
Cascade RCNN	0.208	0.290	0.237	0.001	0.101	0.396
CenterNet	0.184	0.274	0.197	0.012	0.101	0.345
InternImage	0.254	0.340	0.293	0.0008	0.142	0.433

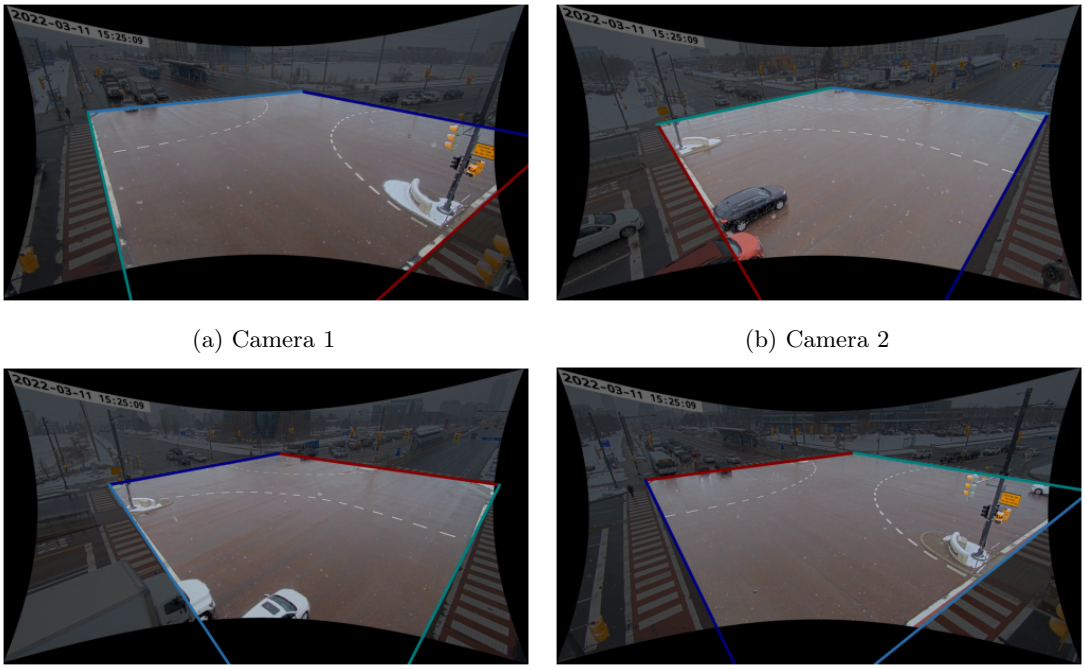
Table 9: Fine-grained performance of fine-tuned detectors on the Region of York dataset.

4.4.2 Detector Performance by Region: Intersection vs. Approaches

The accuracy of vehicle detection may depend on the vehicle's position relative to the intersection. Within intersections, vehicles are clear, more aligned to the camera and less likely to be occluded than in the approaches to the intersection. We partitioned our ground truth data based on the location of detected vehicles. This involved determining whether the midpoint of the bottom edge of each 2D bounding box fell within the intersection area, defined by the RoI (Figure 16). We found that mAP was significantly higher for vehicles within the intersection compared to those in the approaches (Table 10).

Model	Fine-tuned	mAP - Intersection	mAP - Approach
YOLOX	×	0.299	0.038
	✓	0.452	0.152
Cascade RCNN	×	0.333	0.066
	✓	0.495	0.201
CenterNet	×	0.277	0.043
	✓	0.468	0.188
InternImage	×	0.312	0.212
	✓	0.566	0.239

Table 10: Performance of course-grained fine-tuned and pre-trained detectors within and outside the intersection boundaries on the validation set of the Region of York Dataset.



(a) Camera 1

(b) Camera 2

(c) Camera 3

(d) Camera 4

Figure 16: Region of interest polygon for the Region of York dataset.

4.5 Conclusion

Our study shows that detectors have trouble detecting objects in the far field. However, for TMC, the important details are mostly within the intersection area. We found that the InternImage foundation model performed better on our datasets both off-the-shelf and after fine-tuning the detectors. Based on these findings we selected InternImage as our detector for following experiments.

5 Multiple Object Tracking

Multiple Object Tracking (MOT) algorithms track multiple objects across a sequence of frames. In this section, we review general tracking methodologies and detail three specific detectors: SORT[9], DeepSORT[79], and ByteTrack[86].

The MOT problem involves predicting, for each frame t in a sequence, the bounding boxes $B_t = \{b_1, b_2, \ldots, b_N\}$, class labels $L_t = \{l_1, l_2, \ldots, l_N\}$, and tracking IDs $ID_t = \{id_1, id_2, \ldots, id_N\}$. Each bounding box b_i is associated with a specific tracking ID id_i, linking it across the temporal domain [7].

5.1 Background

Detection and tracking can be solved together or one can follow a sequential tracking-by-detection approach in which an independently-trained object detector is used to detect objects in each frame and these are then passed to a tracker [13, 47]. Early tracking research primarily emphasized feature-based methods using particle filters [18] or dynamic feature grouping [36] facilitated by the Kanade-Lucas-Tomasi (KLT) tracker [76]. Additional techniques included region-based tracking [67, 61] and contour-based tracking [82]. Recent trackers utilize a tracking-by-detection approach with a deep object detector, classified into offline and online models based on their adherence to causality in assignment algorithms. Tracktor [60] is a notable offline tracker. Notable online trackers include SORT [72], DeepSort [12, 77, 44, 76], ByteTrack [51], CenterTrack [37], and JDE [84], all employing variations of the Kalman filter. Further enhancements have been made by training re-ID networks (Eg. [14]) to extract unique appearance features to address failures during temporal occlusions [43], and improving tracking precision by integrating trajectories from dual viewpoints [54].

5.2 Method

5.2.1 Causal vs Non-Causal Tracking

Tracking methodologies are categorized into causal (online) and non-causal (offline) based on their use of temporal information. Causal tracking operates frame-by-frame, utilizing only the current and previous frames to determine the association of objects detected in the current frame with those detected in previous frames. This method is suitable for real-time applications but can struggle in complex scenarios without future frame context. Non-causal tracking, on the other hand, considers the entire sequence of frames, allowing for global optimization of the tracking process. Although more accurate and robust, it may not be ideal for real-time applications due to processing latency.

We follow a causal tracking method. Key components of this approach include the motion model, appearance model, and association step [45].

5.2.2 Motion Model

The motion model predicts an object's future location based on its current and previous locations. This is essential for maintaining track of objects when they are temporarily occluded or close to other objects. Linear motion models, suitable for simple and predictable motion patterns, assume constant velocity or acceleration, but their simplicity can be limiting in complex scenarios. Non-linear motion models, such as those based on extended Kalman filters [40] or particle filters [24], offer greater flexibility by accommodating more complex kinematics. However, they often require more computational power and parameter tuning [45].

5.2.3 Appearance Model

An appearance model may help to distinguish different objects, preventing false matches. These models have evolved significantly with technological advancements. Initial models used simple attributes like color histograms, which were robust to occlusion but limited in varied lighting conditions or against similar backgrounds [50, 69, 49, 33]. The advent of deep learning introduced convolutional neural networks that extract deep features, providing a detailed representation of object appearance. The choice between traditional methods and deep feature-based models is often influenced by factors such as computational requirements and the need for extensive, annotated training datasets [45].

5.2.4 Association Model

The association step reconciles new detections with existing tracks by integrating motion and appearance data. The greedy algorithm, known for its computational efficiency, matches detections to tracks in descending order of similarity score. The Hungarian algorithm [39], solves a linear assignment problem that minimizes the mean similarity of matches. Deep learning-based assignment models can identify complex patterns and relationships in the data, enhancing tracking robustness, but require significant computational resources and extensive training data [45]. Thus, selecting an association model necessitates careful evaluation of factors such as the need for real-time processing, scene complexity, and available computational and data resources.

5.2.5 SORT

The paper "Simple Online and Real-time Tracking" [9] presents an efficient approach to multiple object tracking, focusing on real-time, online association of objects across frames. The approach

emphasizes the quality of detection, which significantly influences tracking performance.

The methodology combines two classical techniques: the Kalman Filter for motion prediction and the Hungarian algorithm for data association. For detection, the paper employs the Faster Region CNN (FrRCNN) framework. A linear constant velocity model approximates inter-frame displacements of each object, represented as $x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T$, where u, v represent the horizontal and vertical pixel locations of the object's center, s represents the scale (area), and r represents the aspect ratio of the bounding box. The Kalman Filter updates the target state when a detection is associated, adjusting the velocity components. The Hungarian algorithm assigns detections to existing targets, using an assignment cost matrix based on the IoU distance between each detection and predicted bounding boxes. Assignments are rejected if the detection-to-target overlap is less than a predefined threshold, IoU_{min}.

SORT balances speed and accuracy, operating at 260 Hz, significantly faster than other trackers. Its emphasis on fundamental, efficient techniques and high update rate positions SORT as a robust baseline for real-time tracking. Example of SORT tracks are shown in Figure 18.

5.2.6 Deep SORT

Deep SORT [79] advances SORT by integrating a deep learning-based appearance metric, addressing SORT's vulnerability to identity switches during occlusions and enhancing tracking robustness. The core enhancement is a convolutional neural network trained on a re-identification dataset, introducing appearance information into the tracking algorithm. This is crucial for maintaining accurate track identities in scenarios with occlusions or close interactions, where motion information alone is insufficient. The key technical advancement is the redefined association metric, blending the Mahalanobis distance in the motion space and the cosine distance in the appearance space. A key feature introduced is the matching cascade, prioritizing associations with tracks with more recent detections. Examples of Deep SORT tracks are shown in Figure 19.

5.2.7 ByteTrack

Conventional MOT frameworks discard low-confidence detections, often missing occluded or partially visible objects, leading to fragmented object trajectories. ByteTrack [86], however, includes almost every detection box, enhancing the tracklet association process and significantly improving object recovery in challenging scenarios like occlusions.

Central to ByteTrack's approach is the BYTE (Basic Unit of Tracklet) association method. This

method utilizes both high and low-score detection boxes; Initially, BYTE employs the Kalman filter to predict tracklet locations in the new frame, forming the basis for calculating the similarity between these predicted locations and detection boxes. The similarity metric, IoU or Re-ID feature distances, is crucial in the first stage of association, where high-score detection boxes are matched with tracklets in a greedy fashion. In the subsequent stage, BYTE addresses unmatched tracklets, pairing them with low-score detection boxes using a similar similarity metric. Examples of BYTETrack tracks are shown in Figure 17.



(a) Frame 90 (b) Frame 155

Figure 17: Example tracks generated by ByteTrack on the Region of York dataset.



(a) Frame 90 (b) Frame 155

Figure 18: Example tracks generated by SORT on the Region of York dataset.



(a) Frame 90 (b) Frame 155

Figure 19: Example tracks generated by Deep SORT on the Region of York dataset.

5.3 Evaluation

Tracking evaluation focuses on the accuracy and consistency of object trajectories. Metrics used include Multiple Object Tracking Precision, Multiple Object Tracking Accuracy, and IDF1 score [8].

- **Multiple Object Tracking Precision (MOTP):** Measures the precision of object localization, quantified as the average distance between true and hypothesized positions.

$$\text{MOTP} = \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{11}$$

  where c_t denotes the number of matches in frame t and d_t^i is the bounding box overlap of target i with its assigned ground truth object.

- **Multiple Object Tracking Accuracy (MOTA):** Measures overall accuracy performance using an event-based computation of how often mismatch occurs between the tracking results and ground-truth. MOTA contains the counts of FP (false-positive), FN (false negative), and id-switches (IDSW) normalized over the total number of ground-truth (GT) tracks.

$$\text{MOTA} = 1 - \frac{\sum_t (FP_t + FN_t + IDSW_t)}{\sum_t GT_t} \tag{12}$$

- **IDF1:** To compute an ID-based performance metric, identities from ground truth must first be matched to detected identities [63]. This is achieved by constructing a bipartite graph $G = (V_T, V_C, E)$, where V_T and V_C represent nodes for ground truth and computed trajectories, respectively. V_T is constructed to contain "regular" nodes τ for each ground truth trajectory, along with "false positive" nodes f_γ^+ for each unmatched computed trajectory γ. Similarly, V_C includes regular nodes γ for each computed trajectory and "false negative" nodes f_τ^- for each unmatched ground truth trajectory τ.

  Edges are added to connect each pair (τ, γ) where the ground truth and computed trajectories overlap in time. Additionally, each regular ground truth node τ is connected to its false negative counterpart f_τ^-, and each computed node γ to its false positive counterpart f_γ^+. The cost of each edge (τ, γ) reflects the number of mismatched frames if that match were selected, with the cost function defined in Equation 13 where T_τ and T_γ denote the sets of frames in which trajectories τ and γ are active, IoU_{th} is the IoU threshold used for matching bounding boxes, and $m(\tau, \gamma, t, IoU_{th}) = 1$ if a mismatch occurs at time t (i.e., IoU $< IoU_{th}$ between computed and ground truth bounding boxes) and 0 otherwise.

$$c(\tau, \gamma, IoU_{th}) = \sum_{t \in T_\tau} m(\tau, \gamma, t, IoU_{th}) + \sum_{t \in T_\gamma} m(\tau, \gamma, t, IoU_{th}) \tag{13}$$

For edges between a computed trajectory γ and its false positive node f_γ^+, and between a ground truth trajectory τ and its false negative node f_τ^-, the cost is set as the total number of frames in the unmatched trajectory. This accounts for all frames in γ or τ as false positives or false negatives, respectively, when no match is available.

After minimizing the total edge costs, True Positives (IDTP), False Positives (IDFP), and False Negatives (IDFN) are determined. Specifically, each matched pair (τ, γ) is counted as a True Positive. Each unmatched computed trajectory γ connected to its false positive node f_γ^+ is counted as a False Positive, and each unmatched true trajectory τ connected to its false negative node f_τ^- as a False Negative.

IDFN, IDFP, and IDTP are given in Equations 14, 15, and 16 respectively, where A_T and A_C denote the sets of all ground truth and computed identities included in the matched trajectories and $\text{len}(\tau)$ represents the number of frames in trajectory τ. $\gamma_m(\tau)$ denotes the computed trajectory matched to the ground truth trajectory τ, and $\tau_m(\gamma)$ denotes the ground truth trajectory matched to the computed trajectory γ.

$$IDFN = \sum_{\tau \in A_T} \sum_{t \in T_\tau} m(\tau, \gamma_m(\tau), t, IoU_{th}) \tag{14}$$

$$IDFP = \sum_{\gamma \in A_C} \sum_{t \in T_\gamma} m(\tau_m(\gamma), \gamma, t, IoU_{th}) \tag{15}$$

$$IDTP = \sum_{\tau \in A_T} \text{len}(\tau) - IDFN = \sum_{\gamma \in A_C} \text{len}(\gamma) - IDFP \tag{16}$$

ID Precision (IDP), ID Recall (IDR), and IDF1, the harmonic mean of IDP and IDR, are given in Equations 17, 18, and 19 respectively.

$$IDP = \frac{IDTP}{IDTP + IDFP} \tag{17}$$

$$IDR = \frac{IDTP}{IDTP + IDFN} \tag{18}$$

$$IDF1 = \frac{2}{\frac{1}{IDP} + \frac{1}{IDR}} = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFP + IDFN} \tag{19}$$

5.4 Results

5.4.1 Evaluation of Detector-Tracker Combinations

In our pipeline the performance of TMC depends on tracking and tracking itself depends on detection. Here we evaluate the tracking performance for various detector-tracker combinations. Specifically, we evaluate four pre-trained and fine-tuned detectors: YOLOX, Cascade R-CNN, CenterNet, and InternImage, when paired with three trackers: Sort, DeepSort, and ByteTrack. This results in 24 unique detector-tracker pairings (4 detectors x 2 tuning-modes x 3 trackers). See Table 11 and 12.

Detector	Tracker	MOTA	MOTP	IDF1
YOLOX	Sort	0.183	0.147	0.292
	DeepSort	0.179	0.152	0.294
	ByteTrack	0.160	0.133	0.284
Cascade RCNN	Sort	0.256	0.161	0.401
	DeepSort	0.252	**0.165**	0.403
	ByteTrack	0.244	0.149	0.413
CenterNet	Sort	0.101	0.114	0.175
	DeepSort	0.100	0.117	0.173
	ByteTrack	0.039	0.086	0.074
InternImage	Sort	**0.311**	0.136	0.478
	DeepSort	0.307	0.139	0.478
	ByteTrack	0.303	0.126	**0.487**

Table 11: 2D tracking results on validation set with pre-trained detectors on Region of York data.

Detector	Tracker	MOTA	MOTP	IDF1
YOLOX	Sort	0.484	**0.185**	0.590
	DeepSort	0.482	**0.185**	0.578
	ByteTrack	0.460	0.160	0.614
Cascade RCNN	Sort	0.535	0.143	0.655
	DeepSort	0.533	0.142	0.658
	ByteTrack	0.541	0.136	0.693
CenterNet	Sort	0.570	0.168	0.652
	DeepSort	0.568	0.167	0.648
	ByteTrack	0.521	0.152	0.655
InternImage	Sort	0.581	0.126	0.698
	DeepSort	0.579	0.125	0.700
	ByteTrack	**0.584**	0.119	**0.730**

Table 12: 2D tracking results on validation set with course-grained fine-tuned detectors on Region of York data.

Our findings show improvements in tracking metrics, particularly when trackers are integrated with fine-tuned detectors, illustrating the impact of detector optimization on tracking performance. Additionally, results indicate that trackers paired with InternImage generally outperform others. However, the efficacy of combinations varies by specific metrics: combinations like YOLOX and CenterNet paired with Sort excel in MOTP, while InternImage paired with ByteTrack demonstrates superiority in the IDF1 score. In our application metrics such as MOTP that place emphasis on the precision of the bounding box are less important compared to those such as IDF1 that focus on consistency of IDs since ID switches will directly affect the total number of unique vehicles. Based on this finding we decided to use InternImage paired with ByteTrack, the combination with the highest IDF1 score, as our detector and tracker in our TMC pipeline. The rest of the experiments discussed in the next sections were done using this selection.

5.4.2 Tracking Performance by Region: Intersection vs. Approaches

The accuracy of tracking vehicles can vary based on where the vehicle is located. When the traffic light turns green, vehicles often speed up, which can break the constant speed assumption used

in the Kalman filter and cause tracking errors, leading to the same vehicle getting multiple IDs. On the roads leading to the intersection, vehicles might be hidden behind others, increasing the likelihood of ID switches.

In our TMC pipeline, we focus only on vehicles that pass through the intersection. Vehicles that don't enter the intersection don't impact our TMC results. We determine which tracks, both ground-truth and tracker results, go into the intersection by checking if the middle of the bottom edge of any bounding box in the track is inside the RoI (Figure 16) and label these "Intersection" tracks. All other tracks are labelled "Approache". Table 13 shows tracking performance of pre-trained and fine-tuned detectors paired with trackers separately for tracks that intersect the ROI vs those that do not, on validation partition of the Region of York dataset.

Trackers consistently show better MOTA and IDF1 scores within the intersection than on the approaches. The best performance within intersections and on the appraoches was observed with fine-tuned InternImage and ByteTrack for IDF1 and MOTA.

Fine-tuned detectors improved IDF1 and MOTA scores significantly both within intersections and on the approaches, with the exception of DeepSort, which yielded lower MOTA scores. DeepSort's IDF1 scores improved less than SORT and ByteTrack, suggesting kinematics-based trackers benefit more from detector fine-tuning than appearance-based trackers. This is likely because DeepSort relies on pre-trained deep appearance features as well as kinematics and these are not as readily adapted.

Detector	Tracker	Fine-tuned	Intersection/Approache MOTA	MOTP	IDF1
YOLOX	SORT	×	0.434/ 0.033	0.147/ 0.161	0.558/ 0.080
		✓	0.731/ 0.275	0.148/ 0.210	0.785/ 0.489
	DeepSORT	×	0.414/ -0.215	0.150/ 0.059	0.545/ 0.015
		✓	0.694/-0.310	0.140/ 0.105	0.754/ 0.071
	ByteTrack	×	0.400/ 0.018	0.130/ 0.157	0.574/ 0.052
		✓	0.756/ 0.313	0.132/ 0.178	0.850/ 0.475
Cascade RCNN	SORT	×	0.539/ 0.091	0.148/ 0.171	0.654/ 0.235
		✓	0.796/ 0.332	0.130/ 0.141	0.835/ 0.542
	DeepSORT	×	0.512 / -0.260	0.147/ 0.149	0.645/ 0.035
		✓	0.780/ -0.364	0.121/ 0.067	0.846/ 0.077
	ByteTrack	×	0.543/ 0.116	0.136/ 0.166	0.708/ 0.236
		✓	0.838/ 0.399	0.116/ 0.138	0.906/ 0.582
CenterNet	SORT	×	0.277/ -0.010	0.109/ 0.106	0.394/ 0.020
		✓	0.763/ 0.404	0.127/ 0.197	0.815/ 0.583
	DeepSORT	×	0.283/-0.135	0.107/ 0.020	0.415/ 0.001
		✓	0.758/ -0.310	0.117/ 0.110	0.830/ 0.105
	ByteTrack	×	0.088/ -0.46	0.072/ 0.000	0.158/ 0.000
		✓	0.742/ 0.422	0.108/ 0.183	0.840/ 0.583
InternImage	SORT	×	0.615/ 0.157	0.131/ 0.132	0.728/ 0.334
		✓	0.838/ 0.409	0.122/ 0.124	0.870/ 0.596
	DeepSORT	×	0.589/ -0.279	0.129/0.030	0.717/ 0.052
		✓	0.825/ -0.352	0.112/ 0.030	0.890/ 0.091
	ByteTrack	×	0.613/ 0.185	0.116/ 0.128	0.759/ 0.349
		✓	0.869/ 0.457	0.104/ 0.123	0.926/ 0.631

Table 13: Performance of course-grained pre-trained and fine-tuned detectors paired with trackers for tracks within and outside the intersection on the validation set of the Region of York Dataset.

5.4.3 Tracking Performance by Weather Condition

The performance of multi-object tracking is closely tied to perception and illumination conditions. We benchmarked our best detector–tracker pair across multiple weather and lighting scenarios in the Region of York dataset. Table 14 shows that the system achieves its highest IDF1 (0.780) and MOTA (0.647) under dawn/dusk conditions. Fog also demonstrates relatively strong performance (IDF1 of 0.768, MOTA of 0.625), followed by rain (0.754, 0.603) and cloudy (0.711, 0.564) scenarios. Snowy conditions pose the greatest challenge, yielding the lowest IDF1 (0.671) and MOTA (0.534).

Condition	IDF1	MOTA	MOTP
Snowy	0.671	0.534	0.124
Cloudy	0.711	0.564	0.112
Rain	0.754	0.603	0.154
Fog	0.768	0.625	0.139
Dawn/Dusk	0.780	0.647	0.109

Table 14: Performance of fine-tuned InternImage and ByteTrack over the validation set of the Region of York dataset.

5.5 Conclusion

Since our approach is tracking-by-detection, the quality of detection greatly affects tracking performance both near the camera (intersection) and farther away (approaches).

In terms of IDF1, which is the critical performance metric for TMC, pre-trained InternImage outperformed all other pre-trained detectors, and fine-tuned InternImage outperformed all other fine-tuned detectors. While fine-tuning InternImage on the Region of York training partition led to substantial improvement on the Region of York validation partition, it is unclear whether these improvements would generalize to all other datasets, and whether the fine-tuned InternImage detector is over-fitting to the Region of York dataset. For this reason, we will employ the pre-trained version of InternImage and ByteTrack on the experiments that follow.

6 Single-Camera Trajectory Classification

6.1 Background

Methods for counting tracks according to their trajectories can be broadly categorized into line-based, zone-based, field map-based, prototype matching, or a combination thereof. Line-based methods [54, 60] classify trajectories based on their intersections with predefined virtual lines. Variations of entry and exit zones are utilized in [23, 76, 12, 14], while [76] incorporates an additional central zone to handle shorter tracks. The approach of [37, 84] involves creating proximity and completeness 2D field maps, derived from manually drawn typical movements for each object of interest. Prototype matching techniques define a common trajectory for each object, either manually [25, 44], through semi-automated processes using line or zone intersections [43, 1], or via unsupervised methods like clustering [43, 72]. Subsequent matching of query trajectories employs similarity metrics, comparing them to all prototypes. These metrics include directional measures, such as the cosine distance of trajectory directions, and shape-based measures like the Hausdorff distance [72], longest common subsequence (LCSS) [66], nearest neighbor [1], or combinations thereof [25, 44, 77, 43]. Some studies also explore a cascaded approach, combining zone-based and prototype matching techniques [64].

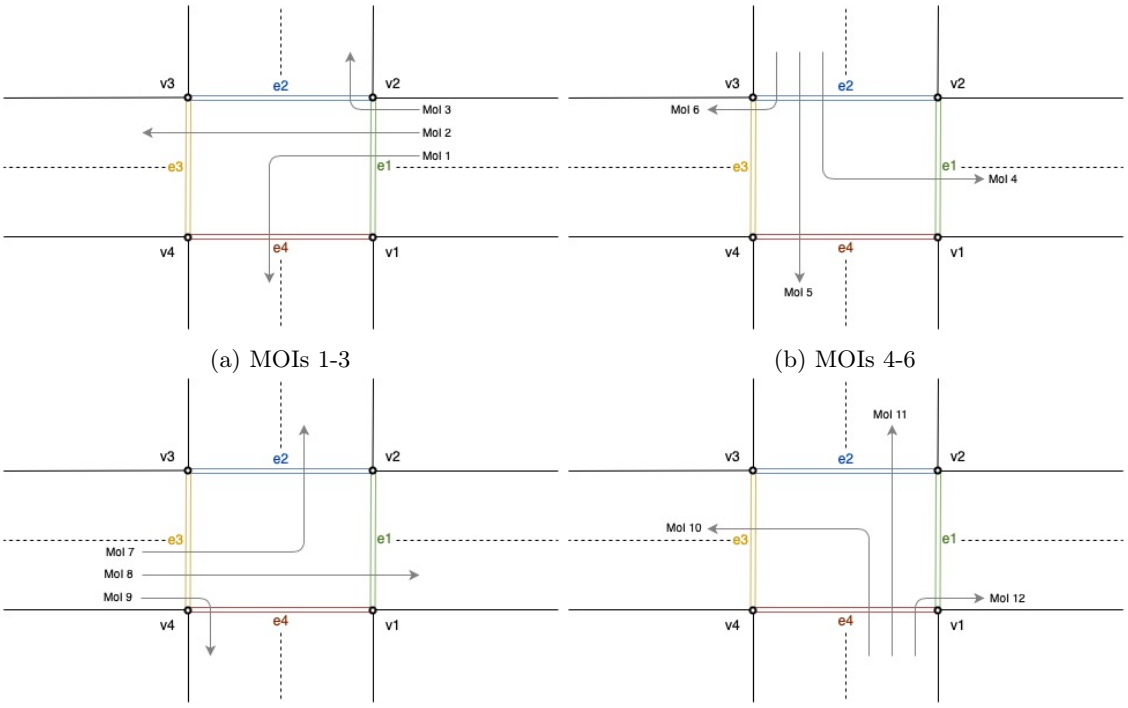
6.2 Method

6.2.1 Movements of Interest

Movements of interest (MoI) are defined by trajectories entering the intersection from one approach and exiting through another. They are characterized by the polygon edge they cross on entry and exit(Table 15, Figure 20).

Entry	Exit	MOI#
e_1	e_4	1
	e_3	2
	e_2	3
e_2	e_1	4
	e_4	5
	e_3	6
e_3	e_2	7
	e_1	8
	e_4	9
e_4	e_3	10
	e_2	11
	e_1	12

Table 15: Assignment of MoI numbers based on entry and exit edges of ROI.



(a) MOIs 1-3

(b) MOIs 4-6

(c) MOIs 7-9

(d) MOIs 10-12

Figure 20: Vertices (v_i) and edges (e_i) of a 4 approach intersection and the 12 MoIs.

6.2.2 Vehicle Trajectory

A track T is defined as a sequence of coordinates over time (Equation 20). For an image-plane track, (x_i, y_i) represents the coordinates of the middle bottom point of a bounding box in frame i, chosen because it approximates the vehicle's contact point with the road. For ground-plane tracks, (x_i, y_i) refers to a coordinate pair on the orthophoto, obtained by back-projecting the image plane coordinate using a homography (Figure 21). N represents the total number of frames where the object is detected.

$$T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \tag{20}$$

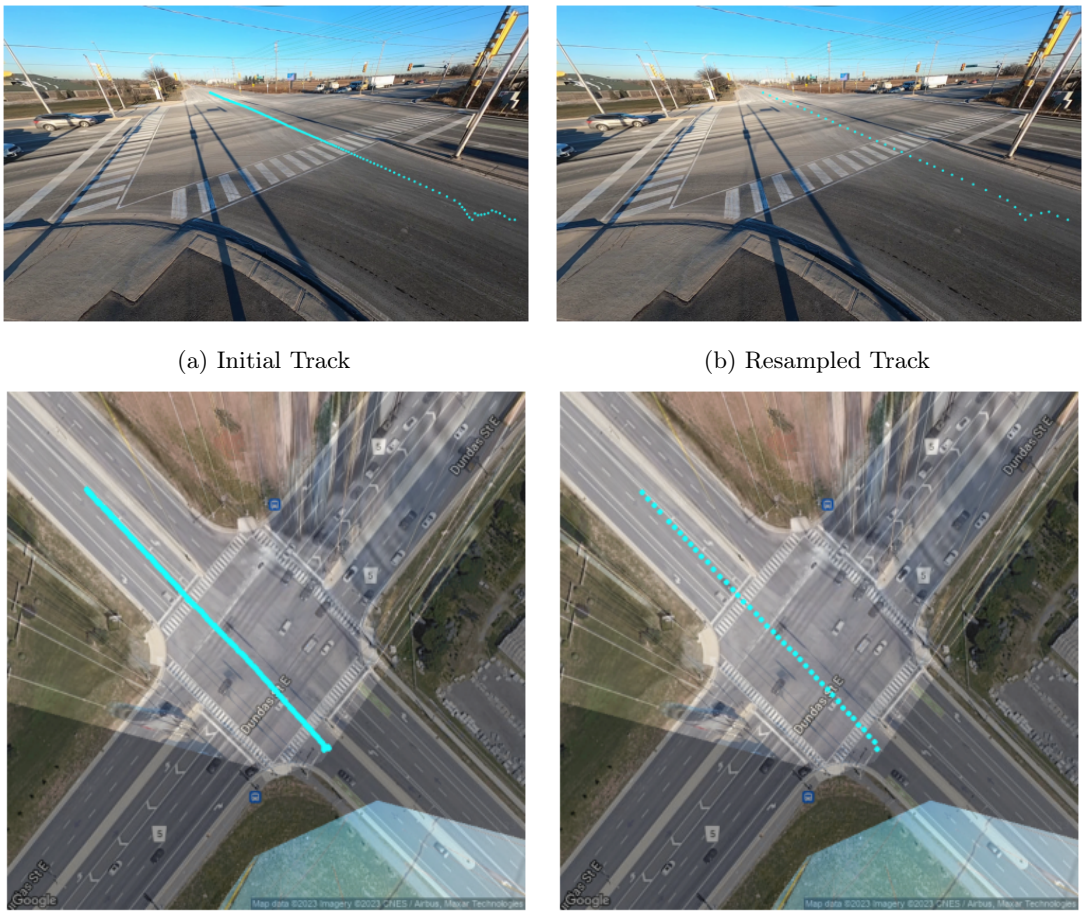


(a) Camera view (b) Orthophoto

Figure 21: Contact point of vehicles marked in both camera view and ground-plane.

6.2.3 Trajectory Resampling

Detections are provided for each video frame (e.g., 60 frames per second), making them uniform over time. This results in non-uniform spatial samples depending on vehicle speed. When classifying a trajectory into MoIs, spatial information is crucial while temporal information is not. The resampling process makes tracks spatially uniform by selecting points so that each has an equal distance from its neighbors (Figure 22). The algorithm starts with the first point and continues through the sequence, selecting points at regular length intervals of 50 pixels in the image and 2 meters on the ground-plane, using linear interpolation.



(a) Initial Track (b) Resampled Track

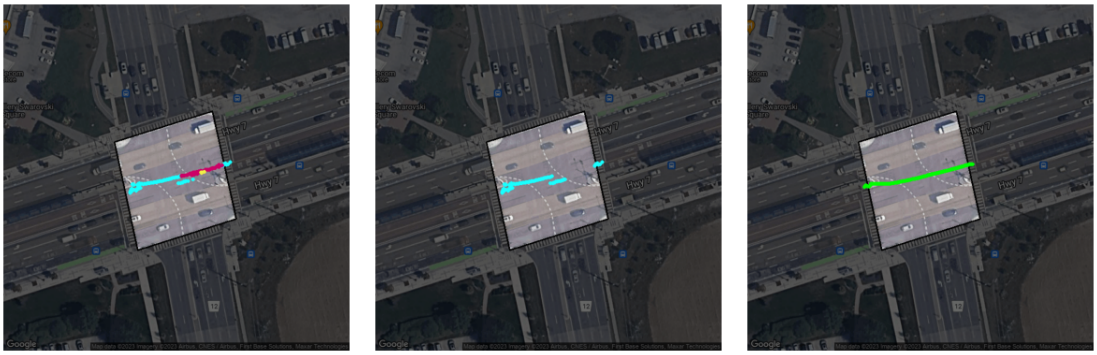
(c) Initial Track (d) Resampled Track

Figure 22: Resampled tracks in camera (a-b) and ground plane (c-d) domain.

6.2.4 Trajectory Filtering

The number of detected tracks impacts our TMC pipeline's performance. ID switches or occlusions can cause trackers to create multiple track fragments, which increases the track count and reduces TMC accuracy. To tackle this, spatial and temporal constraints are often used to remove track fragments. In [44], tracks that are entirely outside the RoI or have a short duration are discarded. Similarly, [37] applies constraints using a 2D completeness field map and sets a minimum travel distance to eliminate interrupted track fragments.

We filtered tracks based on their entry to and exit from the RoI (Figure 23). We discard all tracks that do not exit the RoI (Figure 23 - (a) red and yellow), except for those extending to the last frame of the video. This filtering reduces the impact of track fragmentation on the classification process.



(a) 3 fragments of a track (b) Filtered fragments (c) Ground truth

Figure 23: Example of fragments of a track differentiated by color before (a) and post (b) filtering. The tracks were generated using the pre-trained InternImage detector and ByteTrack on the validation partition of the Region of York dataset. The highlighted quadrilateral indicates the RoI.

6.2.5 Contour Mapping Measure (CMM)

This metric is derived from elastic matching methods which align contours by identifying a mapping between points on these contours that minimizes a matching cost [53]. It aims to find the optimal monotonic mapping from a track $T = (t_1, t_2, ..., t_n)$ to another track $P = (p_1, p_2, ..., p_m)$. The process begins by defining a set of possible mappings between points in T and P. Each mapping is denoted by $s : t \to p$, and the sequence of mappings $S = s_1 s_2 ... s_n$ represents a complete mapping from T to P, ensuring that all points in T are mapped to at least one point in P. Note that if $t_i \to p_k$ and $t_j \to p_l$ then $j > i \to l \geq k$. In other words, the mapping is constrained to be monotonic. The cost of a mapping sequence is given by $\gamma(S) = \sum_{i=1}^{n} \gamma(s_i)$, where $\gamma(s_i)$ is the Euclidean distance between the matched points. The mapping distance $\delta(T, P)$ is defined as the minimum cost of mapping T to P, and can be expressed as:

$$\delta(T, P) := \min_{s} \gamma(S) \tag{21}$$

The normalized mapping distance or the CMM between T and P is given in Equation 22 where $\delta([T], [P])$ is the minimum mapping distance.

$$CMM(T, P) = \frac{1}{n} \cdot \delta(T, P) \tag{22}$$

6.2.6 Turning Movement Classification

Our classification strategy is to use training data to automatically select training tracks that are representative of each different turning movement. At inference, a new track is then compared to these training data to determine its movement class. To define training tracks we use a fully automatic version of a semi-automatic approach that has been used in previous systems [43].

Specifically, we define a training track as a track that enters the ROI through one of its four defining line segments and exits through another (Figure 24 - (a)).

The class of the track is then determined by the entrance and exit segment, which defines $4 \times 3 = 12$ unique turning movement classes. Having defined the training tracks for each turning movement class, we explore two different strategies for summarizing the geometry of these tracks for each class to support efficient inference. In one approach we selected prototypes from the training data and in the other we developed likelihood functions.

6.2.7 Single-Camera Prototype Selection

Prototypes are designed to represent the typical behavior of an object navigating through a specific movement of interest and can be defined either manually or via automated algorithms. These prototypes can take various forms, including hand-drawn sketches, fitted curves, selected portions of recorded tracks, and probability maps.

Our strategy follows a standard approach [77, 64, 72] in which each lane of each turning movement is represented by a single representive prototype track. One novelty of our approach is that we select these prototypes in an unsupervised way. Specifically, we first cluster the training tracks within each movement class using the k-means++ algorithm [30], where the number of clusters k is equal to the number of lanes within the movement class, using RoI boundary crossing locations as feature vectors for the clustering algorithm (Figure 24 - (b)). We then select from each cluster the training track that minimizes the average distance to other training tracks within the cluster, using CMM as a distance metric [53]. This process yields lane-specific prototypes for each turning movement of interest (Figure 24 - (c)).

Figure 25 shows examples of prototypes extracted from all three datasets. For certain intersections like CityFlow S6C42, S6C43 and S6C45 (Figure 25 (h) dark red, (i) dark blue, and (k) orange) prototypes are not well-aligned with the road. This occurs because of inadequate key points needed to solve the homographies and increased sensitivity to these errors further from the camera.

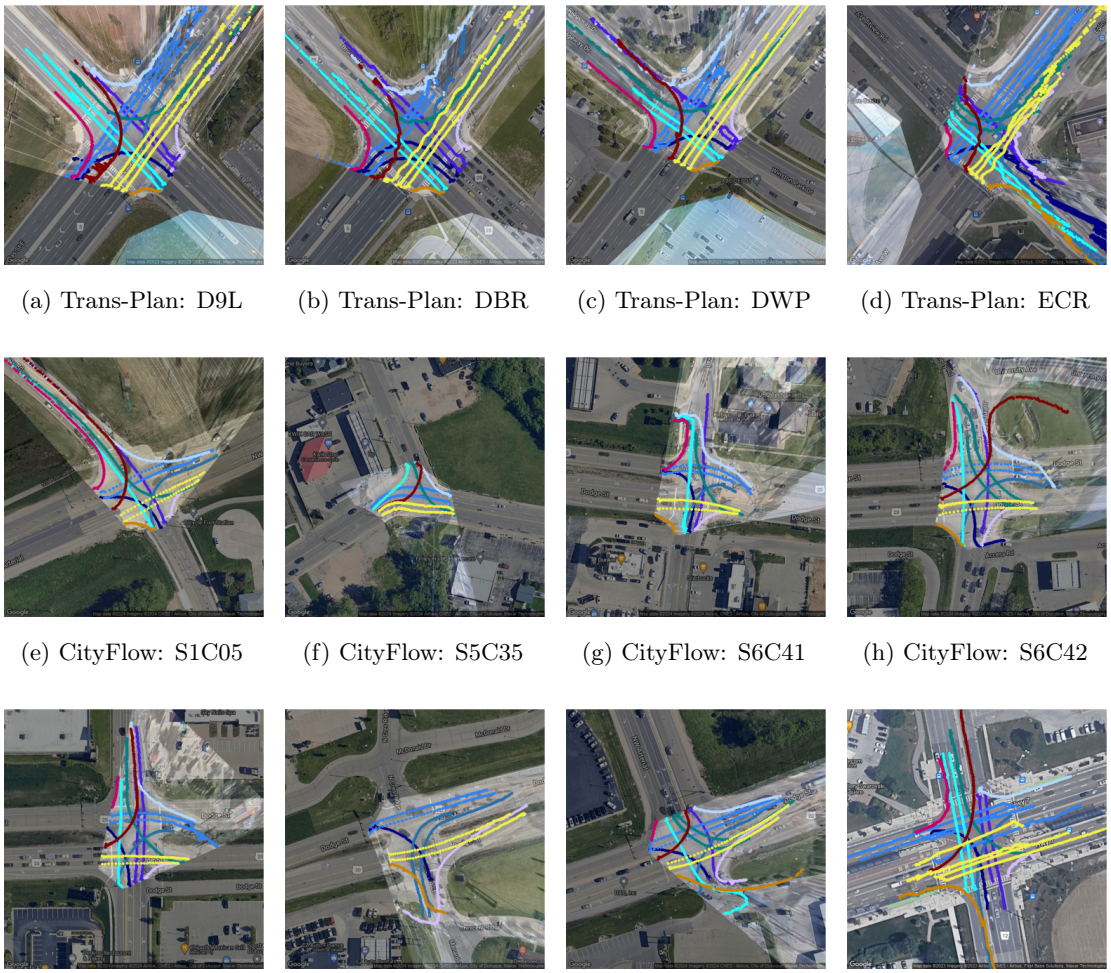
In some instances, prototypes like those from CityFlow S6C42 and S6C45 extend beyond intersection boundaries (Figure 25 (h) dark red, and (k) light blue). These anomalies result from unexpected vehicle movements, where the vehicle made a sharp maneuver, and our algorithm selected these tracks as prototypes.

In some cases, prototypes like those from Trans-Plan DBR, Trans-Plan ECR, and CityFlow S6C45 (Figure 25 (b) blue, (d) Anakiwa, and (k) blue) the prototype will follow the FoV lines instead of ending. This happens because the detector will try to provide bounding boxes when the vehicles are entering and exiting the field of view. These bounding boxes will provide inaccurate estimates of the vehicle contact point and hence leaving irregularities on the extremes of some prototypes.



(a) Training tracks (b) RoI boundary crossings (c) Prototypes

Figure 24: Prototype selection - Trans-Plan dataset.



(a) Trans-Plan: D9L (b) Trans-Plan: DBR (c) Trans-Plan: DWP (d) Trans-Plan: ECR

(e) CityFlow: S1C05 (f) CityFlow: S5C35 (g) CityFlow: S6C41 (h) CityFlow: S6C42

(i) CityFlow: S6C43 (j) CityFlow: S6C44 (k) CityFlow: S6C45 (l) Region of York: Train

Figure 25: Movement prototypes selected via our methodology and color marked by MoI.

6.2.8 Trajectory Classification

The objective of TMC is to classify each track T into one of the predefined movement classes, $M = \{m_1, m_2, \ldots, m_k\}$. TMC can be addressed through two principal categories of classifiers: Rule-Based and Discriminative.

- **Entry-Exit (EE) method:** The entry-exit method does not use learned turning movement models but simply classifies an inference track based on its geometry relative to the four line segments bounding the ROI. In particular, inference tracks that cross two different segments are classified exactly as training tracks are classified in Section 6.2.7. If the inference track starts or ends within the ROI, the closest ROI line segment is assigned as the entry or exit segment. Once the entry and exit edges (e_i, e_j) are determined, a predefined turn movement is assigned using Table 15.

- **Direction (DIR) method:** The direction method only uses the start and end points of the inference track. However, unlike the entry-exit method, the direction method makes

use of the prototype tracks selected during the unsupervised learning stage of our system. In particular, the direction method compares the overall direction of the inference track to the overall directions of the prototype tracks, assigning the turning movement class of the prototype with direction that is most similar to the direction of the inference track.

The direction of each prototype track is defined by the unit vector pointing from its entry point to its exit point. For inference tracks that extend across the ROI, direction is defined in the same way. For inference tracks that start and/or end within the ROI, the start and/or endpoint are employed to define the direction (Equation 23). Direction similarity is defined by the dot product of the unit vectors (Equation 24).

$$V_i = T[N] - T[0] \qquad or \qquad V_i = T[index\ of\ exit] - T[index\ of\ entry] \tag{23}$$

$$d(T_i, P_j) = 1 - \frac{\langle V_i, V_j \rangle}{\|V_i\| \|V_j\|} \tag{24}$$

- **Voting (VOTE) method:** Unlike EE and DIR that only use two points on the inference track, our VOTE method examines the entire track. It marches over the points of the inference track, for each point casting a vote for the turning movement class with the closest prototype to that point. The inference track is then assigned to the movement class with the most votes.

  In detail, for a given inference track $T_i = \{t_1^i, t_2^i, \ldots, t_N^i\}$ we first determine m_n^* the MOI assigned to the point t_n^i with $d(t_n^i, P_j)$ as the Euclidean distance (Equation 25). Then M_i^*, the overall label of T_i, is determined through majority voting (Equation 26).

$$m_n^* = \text{argmin}_j \; d(t_n^i, P_j) \tag{25}$$

$$M_i^* = \text{mode}(\{m_1^*, m_2^*, \ldots, m_N^*\}) \tag{26}$$

- **Maximum Likelihood (ML) method:** Our maximum likelihood method relies upon a learned likelihood model, computing the conditional likelihood of each inference track for each turning movement class and assigning the class with the highest likelihood. While generally the prior distribution over movement classes is far from uniform, we refrain from applying a prior, partly because it is not clear how stable this prior would be over time of day.

  In this strategy, we form a probabilistic model over the distribution of trajectories within each turning movement class. Here we employ Gaussian kernel density estimation (KDE) and

do not rely upon explicit clustering of lanes with each movement class. Our goal is to form a Gaussian KDE model of the conditional likelihood $p\left(T_i|M_j\right)$ of a track T_i given it originates from movement class M_j.

Each track T_i consists of a sequence of n_i 2D positions $T_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}$. We model the likelihood of each of these positions as conditionally independent, so that the conditional likelihood each track T_i is given by Equation 27 where $p\left(\mathbf{x}_{ik}|M_j\right)$ is modeled with a Gaussian KDE.

$$p\left(T_i|M_j\right) = \prod_{k=1\ldots n_i} p\left(\mathbf{x}_{ik}|M_j\right). \tag{27}$$

6.2.9 KDE Bandwidth Optimization

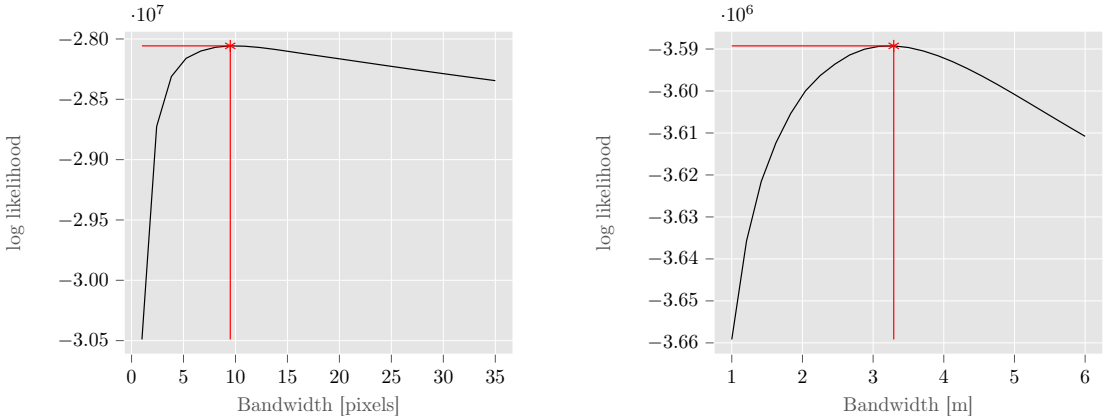
Optimizing the bandwidth parameter for the Guassian KDE model significantly influences the model's accuracy and generalizability. An optimal bandwidth ensures that the KDE captures the true underlying distribution of the data without overfitting to noise.

To optimize this parameter, we used training videos from three datasets. For Trans-Plan and CityFlow, each training video was divided into two equal parts: the first for training the KDE model and the second to assess the accuracy of the model as represented by the likelihood of the held-out data. In the Region of York Dataset, tracks from all training segments for each viewpoint were aggregated and then split similarly.

Figure 26 shows examples of how likelihood on held-out validation data varies with bandwidth for a specific video segment. Table 16 shows how optimal bandwidth varies across segments and datasets. For evaluation of our TMC pipeline we will use the average of these optimal values across datasets; 15.4 pixels in the image and 3.05 meters in the ground-plane. Figures 27, 28, 29, and 30 show the resulting KDE models for the Region of York dataset.

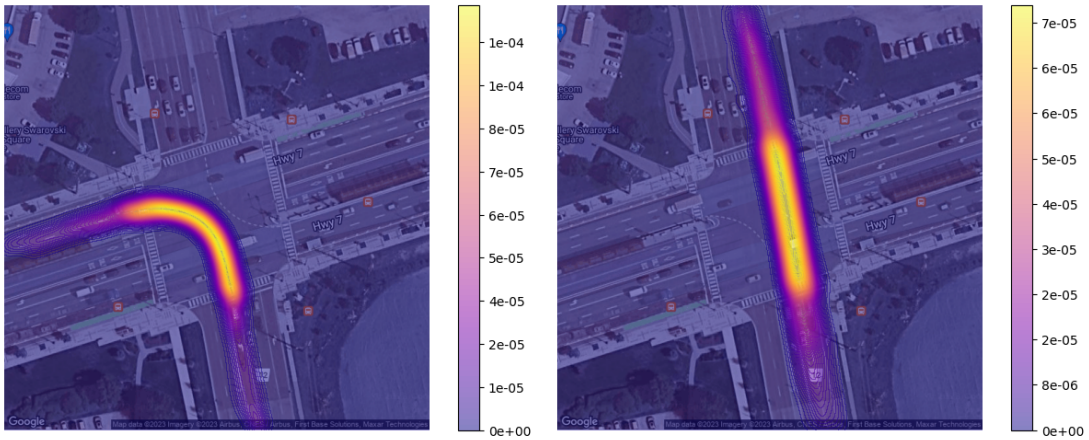
Dataset	Source	Ground plane (meter)	Image plane (piexels)
TransPlan	D9L	3.29	8.0
	DBR	3.29	9.5
	DWP	2.04	8.0
	ECR	3.50	9.5
CityFlowV2	C046	5.16	22.1
	C045	3.29	27.4
	C044	2.04	6.2
	C043	3.70	22.1
	C042	1.83	16.7
	C041	2.25	6.3
	C035	1.56	8.9
	C005	2.45	16.9
Region of York	SC1	3.62	21.1
	SC2	3.62	21.1
	SC3	3.62	21.5
	SC4	3.62	21.1
Avg		3.05	15.4
Std		0.94	7.1

Table 16: Optimal bandwidth for Gaussian kernel density estimation across different camera instances in the three datasets.

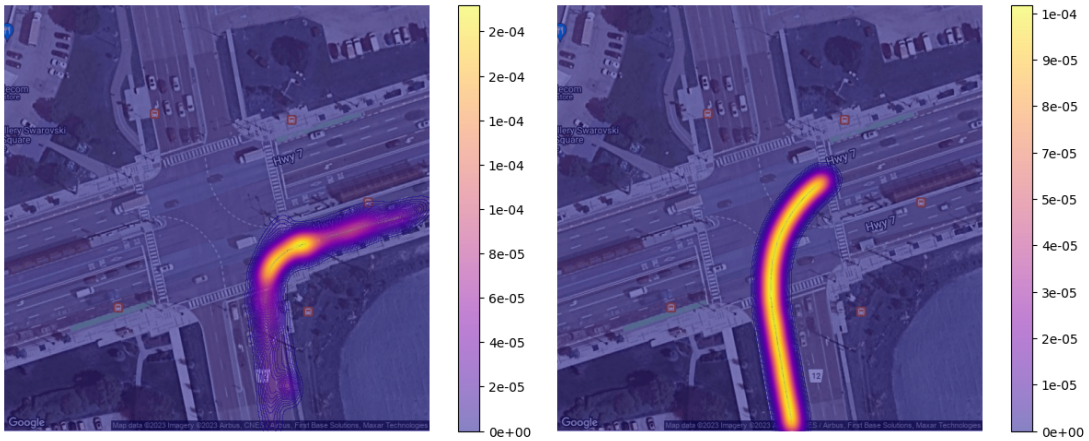


(a) (b)

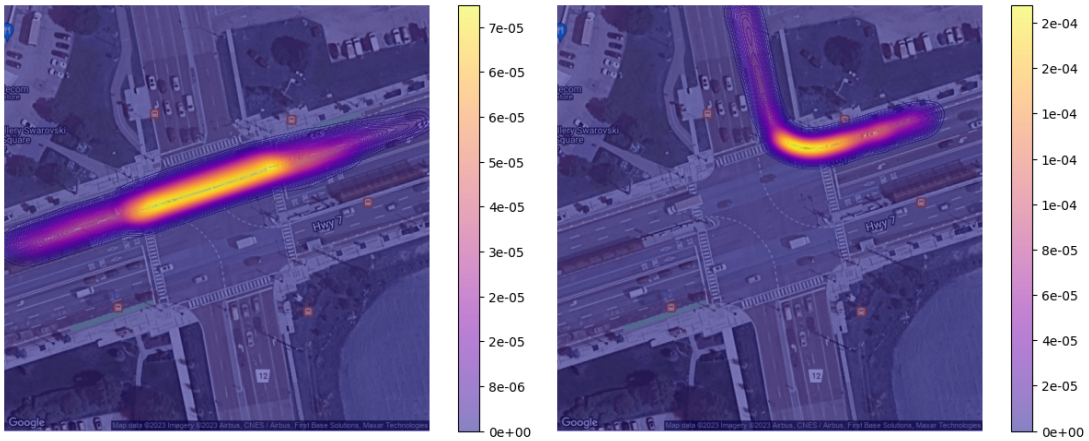
Figure 26: Log probability of held-out validation data (Trans-Plan DBR segment) under a KDE model as a function of the bandwidth of the Gaussian kernel, in the image (a) and ground-plane (b) domain.



(a) MOI 1 (b) MOI 2

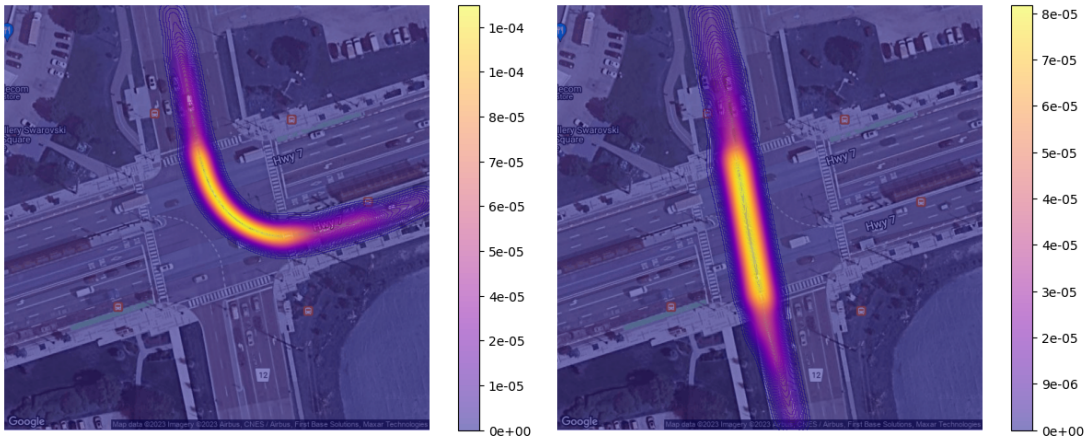


(c) MOI 3 (d) MOI 4



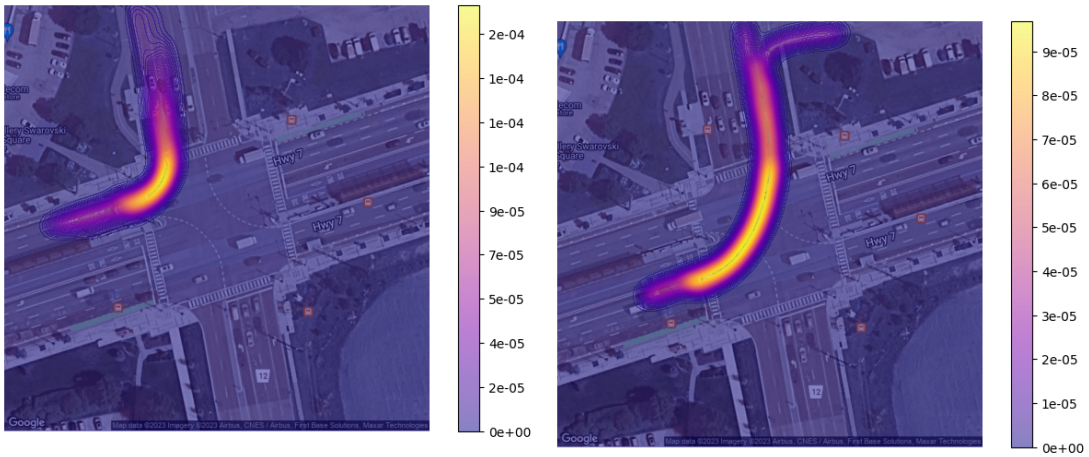
(e) MOI 5 (f) MOI 6

Figure 27: Class conditional density estimations for MoIs 1-6 for the ground-plane model on the Region of York dataset. Bright yellow regions signify areas of higher probability, while dark purple regions indicate areas of lower probability.



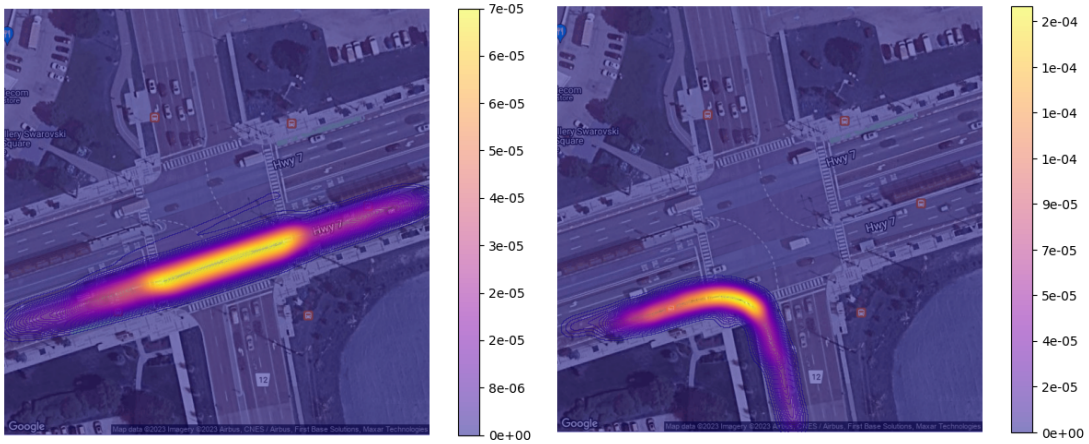
(a) MOI 7

(b) MOI 8



(c) MOI 9

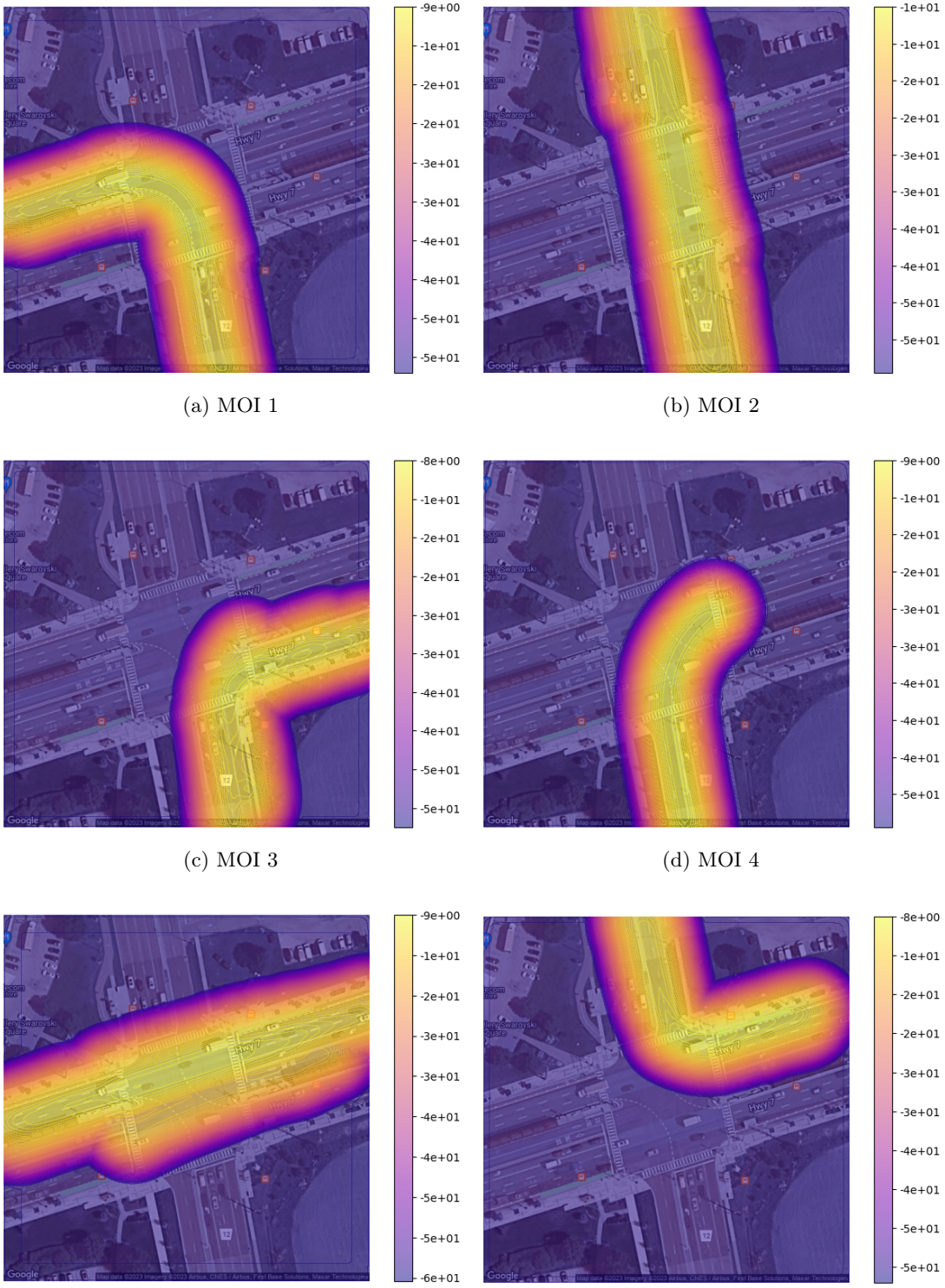
(d) MOI 10



(e) MOI 11

(f) MOI 12

Figure 28: Class conditional density estimations for MoIs 7-12 for the ground-plane model on the Region of York dataset. Bright yellow regions signify areas of higher probability, while dark purple regions indicate areas of lower probability.

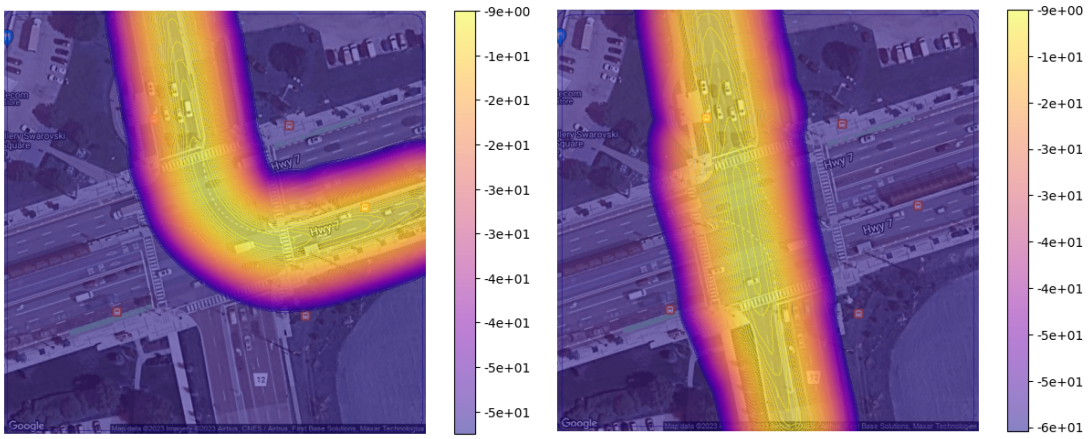


(a) MOI 1 (b) MOI 2

(c) MOI 3 (d) MOI 4

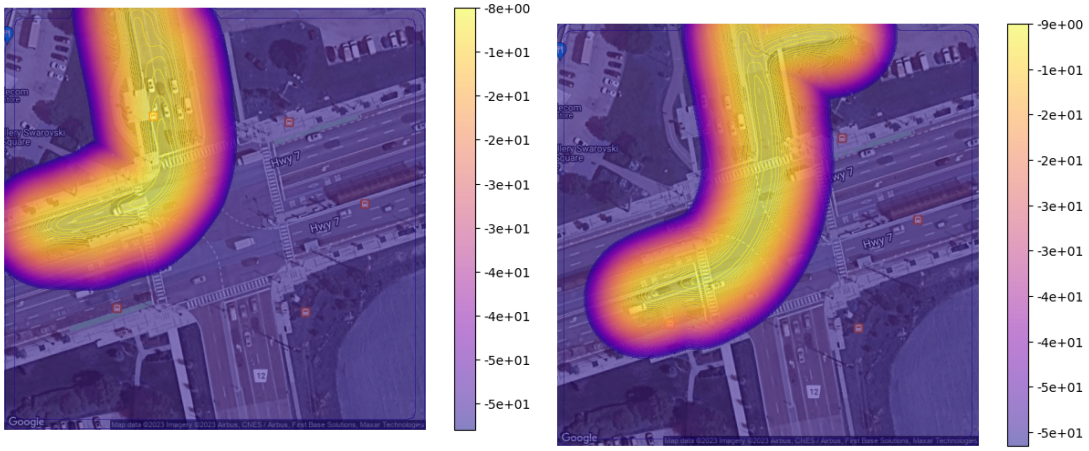
(e) MOI 5 (f) MOI 6

Figure 29: Class conditional log density estimations for MoIs 1-6 for the ground-plane model on the Region of York dataset. Bright yellow regions signify areas of higher probability, while dark purple regions indicate areas of lower probability.



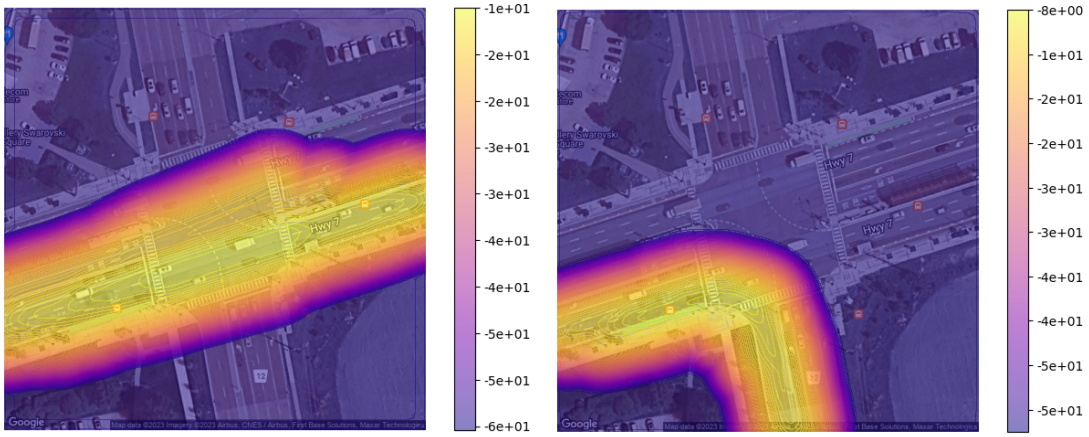
(a) MOI 7

(b) MOI 8



(c) MOI 9

(d) MOI 10



(e) MOI 11

(f) MOI 12

Figure 30: Class conditional log density estimations for MoIs 7-12 for the ground-plane model on the Region of York dataset. Bright yellow regions signify areas of higher probability, while dark purple regions indicate areas of lower probability.

6.3 Evaluation

6.3.1 Single-Camera TMC Evaluation

Turn count classification is assessed by the number of count errors.

$$MAE = \frac{1}{N} \sum_{j=1}^{12} |\hat{n}_j - n_j| \tag{28}$$

where $\hat{n}_j$ and n_j are estimated and ground truth counts for MoI j, respectively and $N = \sum_{j=1}^{12} n_j$ is the total ground truth count.

6.3.2 Single-Camera TMC Bias

Turn count classification bias is assessed by the difference between the estimated and total number of movements.

$$\text{Bias} = \frac{\hat{N} - N}{N} \tag{29}$$

where $\hat{N} = \sum_{j=1}^{12} \hat{n}_j$ is the total estimated count. We only report TMC bias in Section 6.4.3 to provide deeper insight into the cross-dataset generalizability of fine-tuned detectors.

6.4 Results

6.4.1 Camera Plane vs Ground Plane Prototype Selection and Classification

Following the prototype selection methodology explained in Section 6.2.7 we estimated turn counts using the image and ground-plane classifiers discussed in Section 6.2.8. Table 17 compares turn count MAE on the CityFlow dataset and on the validation partitions of our TransPlan and Region of York datasets. We report mean absolute error over all turning movement classes. We evaluate all four of our turning movement count classification methods in both camera and ground-plane domains. We do not report the performance of the EE method on the Region of York dataset because the camera installation lacks full visibility of all intersection entries and exits, making the method inapplicable.

Dataset	Domain	EE	VOTE	DIR	ML
TransPlan	Camera	17.76%	14.24%	12.66%	11.03%
	Ground	14.37%	13.81%	10.55%	**9.93%**
City Flow	Camera	16.09%	7.82%	6.68%	5.18%
	Ground	14.07%	7.91%	5.45%	**4.04%**
Region of York	Camera	———	38.27%	20.00%	19.50%
	Ground	———	38.27%	14.96%	**13.82%**

Table 17: TMC MAE across four classification methods on both camera and ground-plane. We evaluate on the training partition of the CityFlow dataset and on the validation partitions of Trans-Plan and Region of York datasets.

We find that the direction method performs substantially better than the entry-exit method, presumably due to the finer-grained information about trajectory direction provided by the prototypes. Interestingly, despite using the entire trajectory, our voting method performs worse than our direction method. This suggests that the endpoints of the tracks may in fact be more diagnostic of the turning movement class than interior points and should be weighted more heavily.

We find that our maximum likelihood method performs best of all four methods evaluated, on all three datasets. This indicates that, while a voting strategy is insufficient, it can be beneficial to consider the whole track. We note that by summing the log likelihood over all points on the inference track, the ML method allows points that are highly deviant from the conditional model to have more than an equal vote and ultimately overrule inliers.

With only two exceptions we see a consistent improvement when we classify tracks on the ground-plane rather than the image. This makes sense, as projection to the image distorts and potentially entangles the trajectories, and renders stationarity assumptions implicit in distance measurements suspect.

Ultimately our best performance is achieved using our maximum likelihood method on the ground-plane. The superiority of the ML method may be due to the fact that it is the only method that learns from training data, learning a probabilistic model based on the training tracks and optimal kernel bandwidth. The superiority of the ground plane method originates from better and more consistent separation of tracks from different MoIs compared to tracks in the image plane, which are distorted due to projection.

Overall, lowest error rates are achieved on the CityFlow dataset. We believe this is due to the wide field of view, which provides sightlines beyond the boundaries of the ROI, and the relatively high vantage, which minimizes occlusions.

The higher error rates seen in the TransPlan data are likely due to the lower height of the camera, which leads to more occlusions, squeezes the projection of different turning movement classes more closely together due to projection, and increases the error in back-projection due to the high obliquity of the ground plane.

The highest errors are encountered with the Region of York dataset. These arise because of the limited field of view (86 deg horizontal) of the cameras, which means that none of the cameras see the complete ROI.

6.4.2 Impact of Camera Sway

Environmental factors like wind can cause sway in camera. This is particularly evident in the Trans-Plan dataset due to the more flexible pole employed there. To assess the impact of this sway on TMC, we partitioned the test dataset into 3 categories based on a subjective assessment of the severity of camera movement (Table 18). We found that as camera sway increases, the performance of our classification methods declines. This observation highlights the susceptibility of our approach to camera instability.

Sway	**Camera**	**IP**	**VNN**	**Hausdorff**	**Cos**	**kDE**
Minor	D9L	11.69%	6.82%	9.98%	5.84%	4.87%
	DWP	16.23%	16.04%	11.59%	6.21%	8.81%
Moderate	DBR	13.16%	10.38%	18.35%	12.46%	10.38%
Notable	ECR	16.20%	22.18%	16.20%	15.97%	15.51%

Table 18: Turning movement classification percent error of ground-plane counters across different sway conditions of the Trans-Plan dataset.

6.4.3 Impact of Fine-Tuning the Detector

Table 19 gives TMC error rates and biases on the validation set of the Region of York dataset with fine-tuned and pre-trained InternImage and ByteTrack with the ground-plane ML counter.

We found a small improvement in TMC error rate ($< 2\%$) and bias ($< 1\%$) for the Region of York dataset. However, we also found that fine-tuning the InternImage detector on the Region of York dataset reduced TMC performance on the Trans-Plan and CityFlow datasets by 0.67% and 3.32% respectively. This shows that fine-tuning on the Region of York dataset worsens generalization.

Dataset	Fine-tuned	MAE	Bias
Trans-Plan	×	**9.93%**	1.03%
	✓	10.6%	-0.91%
CityFlow	×	**4.04%**	-1.40%
	✓	7.36%	2.81%
Region of York	×	13.82%	5.27%
	✓	**12.34%**	5.06%

Table 19: TMC performance (mean absolute error and mean bias over turning movement classes) of our ground-plane ML counter with pre-trained and fine-tuned systems on the Region of York validation partition.

6.4.4 Impact of Weather and Illumination

The performance of counting depends heavily on the environmental conditions. We benchmark our best single-camera counting method under diverse weather conditions of the Region of York dataset. Table 20 shows that under sunny, clear conditions, the approach achieves its lowest MAE of 3.12%. By contrast, dawn/dusk scenarios with clear illumination but bare roads yield the highest MAE of 33.33%. Cloudy and snowy conditions produce a moderate MAE of 10.34%, while nighttime rain and fog further degrade performance to 18.75% and 30.77%, respectively. There is no clear relationship between the number of tracks or the activity level in a segment and its TMC performance, suggesting that our system is more sensitive to environmental conditions than to traffic volume.

Segment	Illumination	Weather	Road Surface	Activity	# Tracks	MAE
Seg16	Sunny	Clear	Wet	High	32	3.12%
Seg00	Cloudy	Snow	Snow	High	58	10.34%
Seg25	Night	Rain	Wet	Low	16	18.75%
Seg10	Cloudy	Fog	Wet	Low	39	30.77%
Seg26	Dawn/Dusk	Clear	Bare	Medium	42	33.33%

Table 20: TMC MAE of ground-plane ML on the validation partition of Region of York dataset under variable weather conditions.

6.4.5 Failure Modes

Figures 31-40 show several failure modes of our system, which can be classified into three categories. In the first category (Figures 31–34), the inference tracks are fragmented within the intersection boundary. The filtering method discards the initial segment because it does not exit the intersection, while the remaining parts fall within overlapping areas of two MoI densities, leading the system to make incorrect predictions. A potential remedy in this case is to develop a method for reassembling the fragmented tracks.

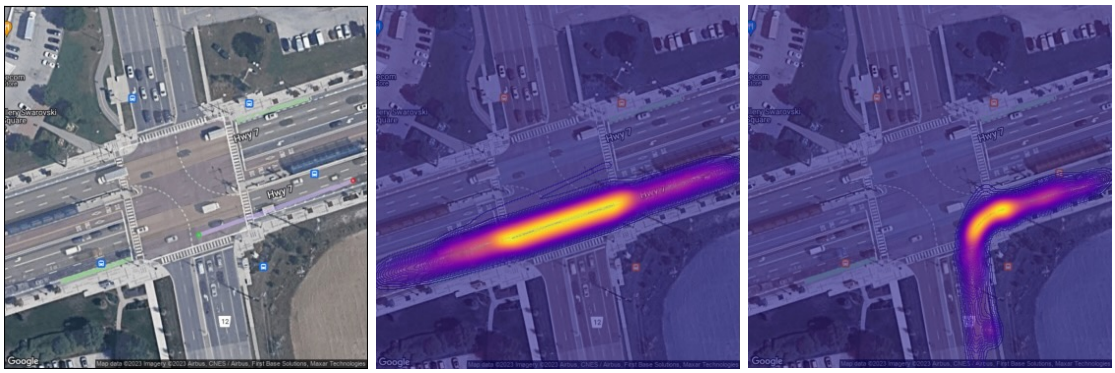
In the second category (Figures 35–38), tracks terminate within the intersection boundaries. This phenomenon occurs only for tracks that end with the video, indicating that the fragmentation results from the conditions of the testing data rather than a tracking error. However, the remaining fragment lies in an area shared by several MOI densities, which again results in ambiguous predictions. Addressing this issue may involve using datasets where videos conclude in a way that prevents vehicle trajectories from being abruptly truncated.

In the final category (Figures 39–40), the inference track is a combination of two MoIs due to re-identification errors in the tracking module. In these cases, the system is presented with a faulty track and predicts the MoI that most likely describes the combined movement. Improving the tracking module to better identify and correct re-identification errors could help mitigate these failures.

One potential cause for the observed failures is the normalization of MOI densities by the total number of points across all prototype tracks, a default property of KDE. This normalization causes longer MOI densities to have lower likelihood values in overlapping areas compared to shorter MOI densities. As a result, if most query points fall within an overlapping region, the system always favors the shorter MOI density. This hypothesis explains the failures illustrated in Figures 31–36. A simple solution is to normalize by the number tracks rather than the number of points.

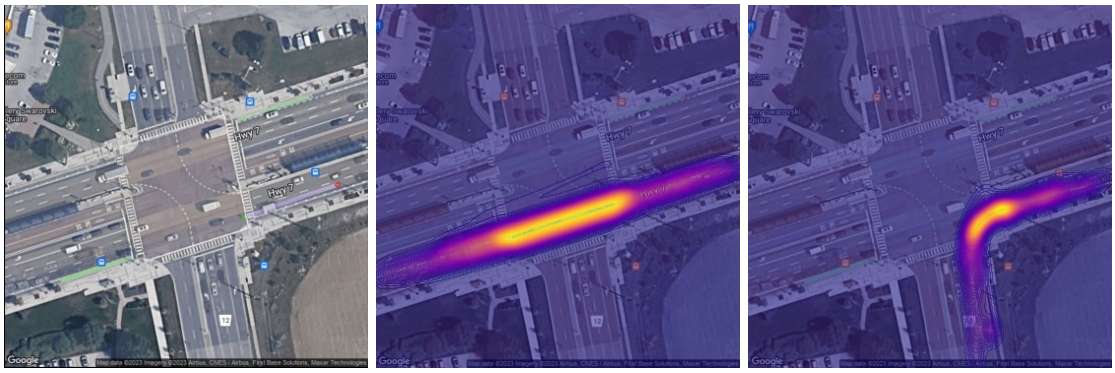
Another possible cause is the assumption that data points within tracks are independent. This assumption neglects sequential relationships between data points. The failure cases shown in Figures 34–37 align with this explanation. These sequential relationships could potentially be captured with a Markov model in which the conditional likelihood of each sample depends both upon the KDE model prior and the location of the previous sample in the track.

Overall, most errors originate in the tracking module and propagate into the counting module. Improvements in tracking accuracy, specifically reducing fragmentation and identity switches, could independently improve overall system performance and allow simpler assumptions in the counting module.



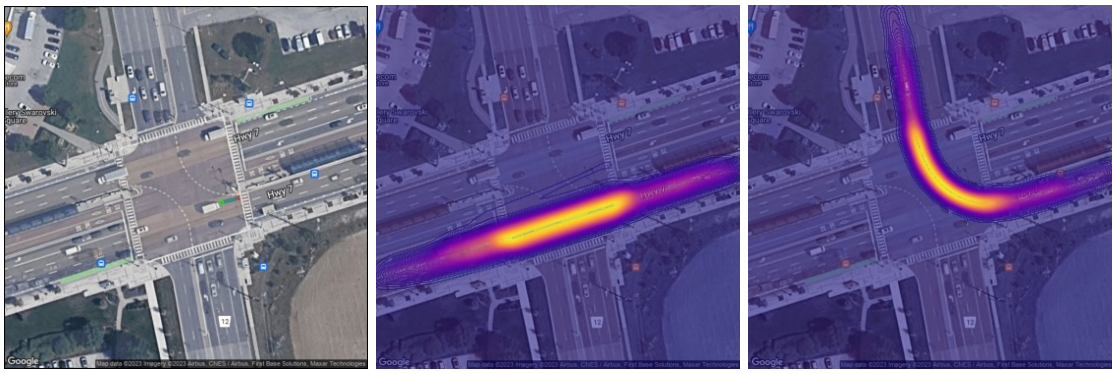
(a) Inference track (b) Density for correct MoI (c) Density for predicted MoI

Figure 31: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 3 (c) instead of the correct value of 11 (b).



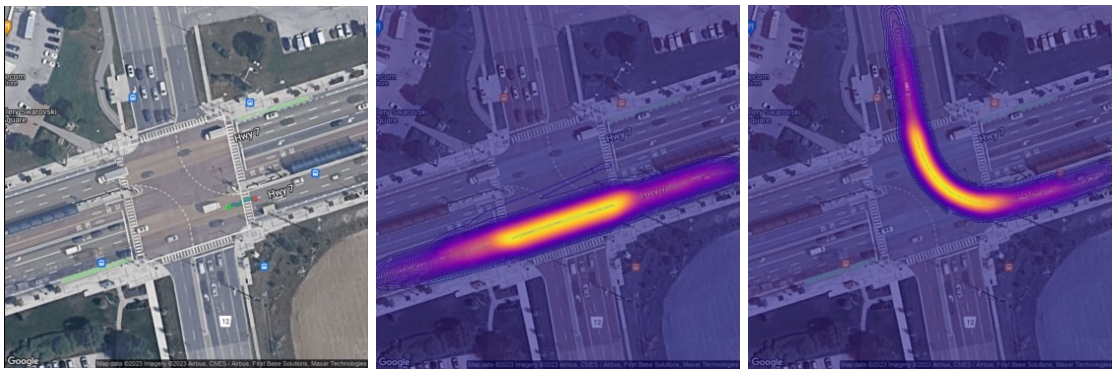
(a) Inference track (b) Density for correct MoI (c) Density for predicted MoI

Figure 32: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 3 (c) instead of the correct value of 11 (b).



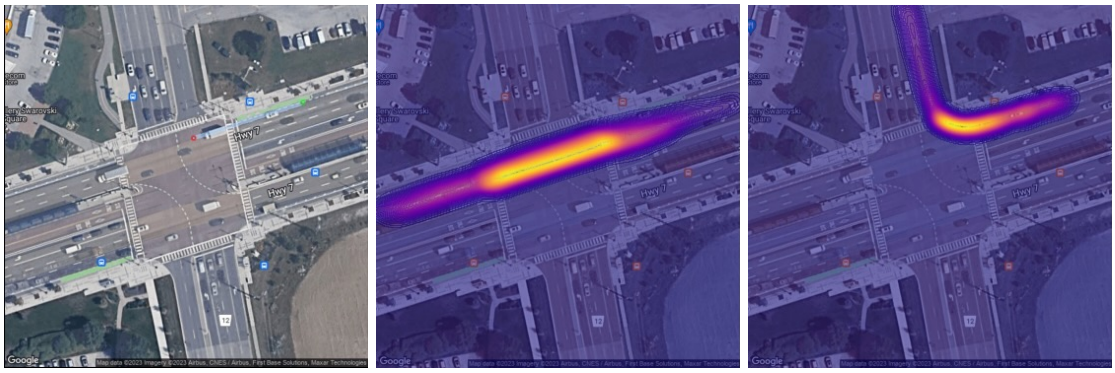
(a) Inference track (b) Density for correct MoI (c) Density for predicted MoI

Figure 33: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 7 (c) instead of the correct value of 11 (b).



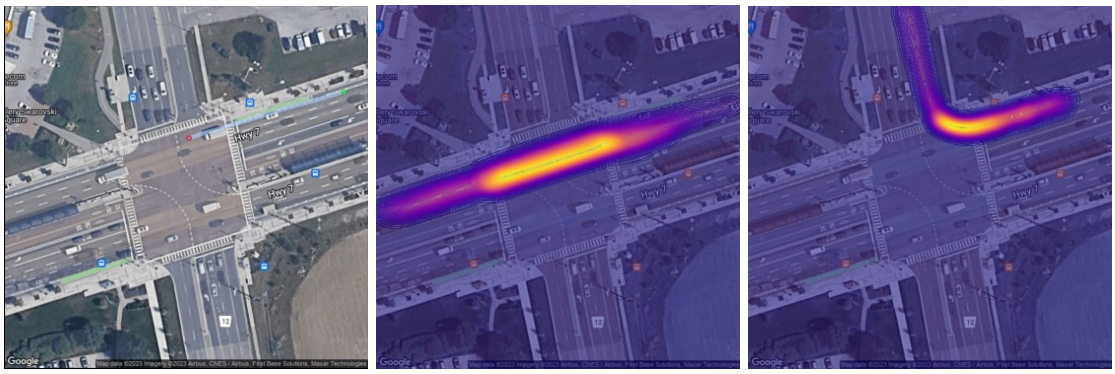
(a) Inference track (b) Density for correct MoI (c) Density for predicted MoI

Figure 34: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 7 (c) instead of the correct value of 11 (b).



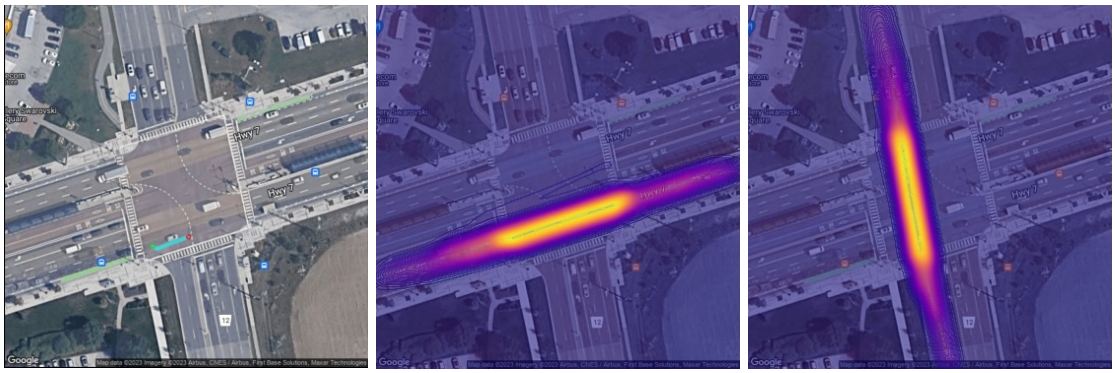
(a) Inference track (b) Density for correct MoI (c) Density for predicted MoI

Figure 35: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 6 (c) instead of the correct value of 5 (b).



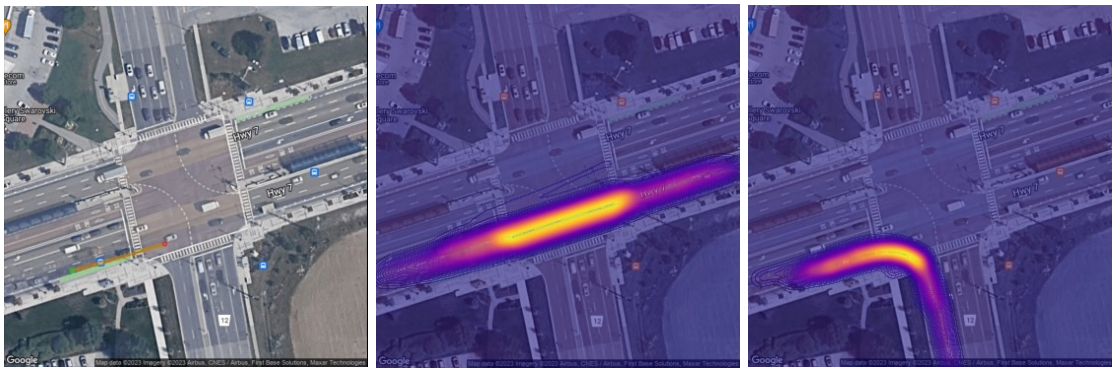
(a) Inference track (b) Density for correct MoI (c) Density for predicted MoI

Figure 36: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 6 (c) instead of the correct value of 5 (b).



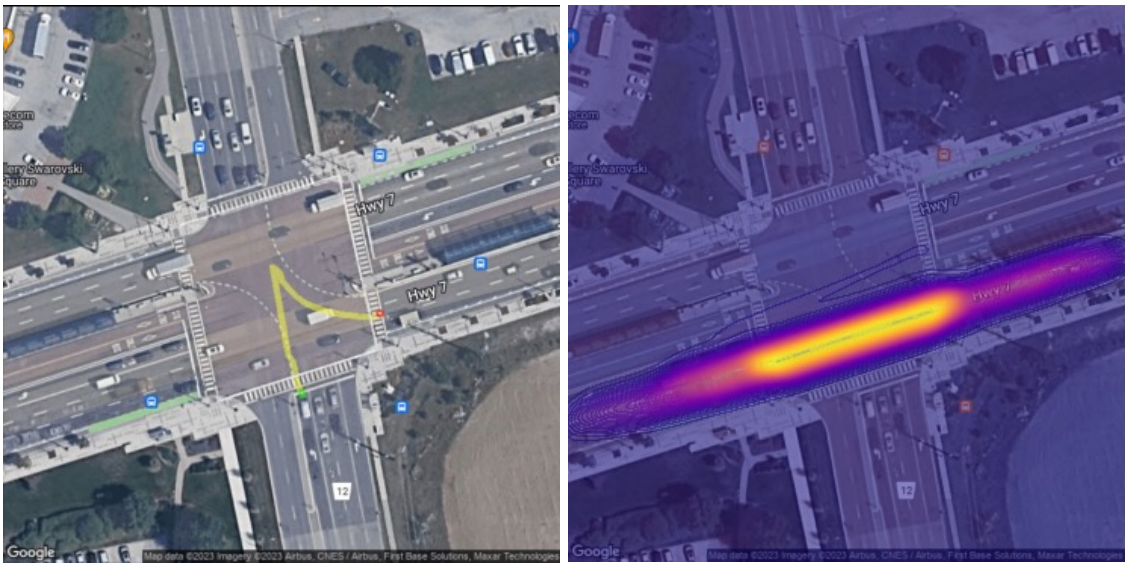
(a) Inference track (b) Density for correct MoI (c) Density for predicted MoI

Figure 37: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 8 (c) instead of the correct value of 11 (b).



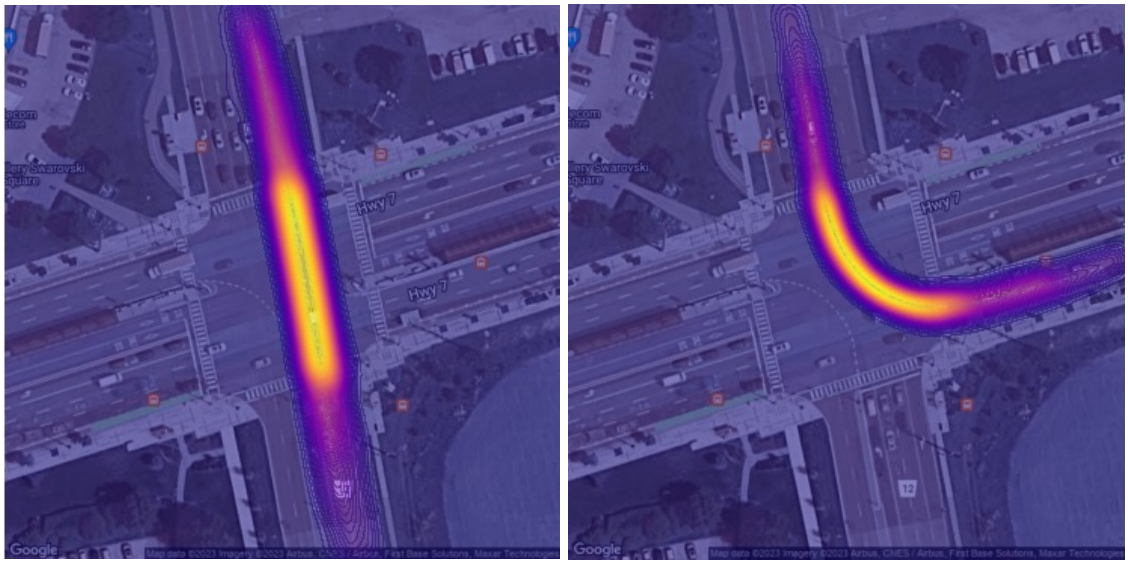
(a) Inference track (b) Density for correct MoI (c) Density for predicted MoI

Figure 38: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 12 (c) instead of the correct value of 11 (b).



(a) Inference track

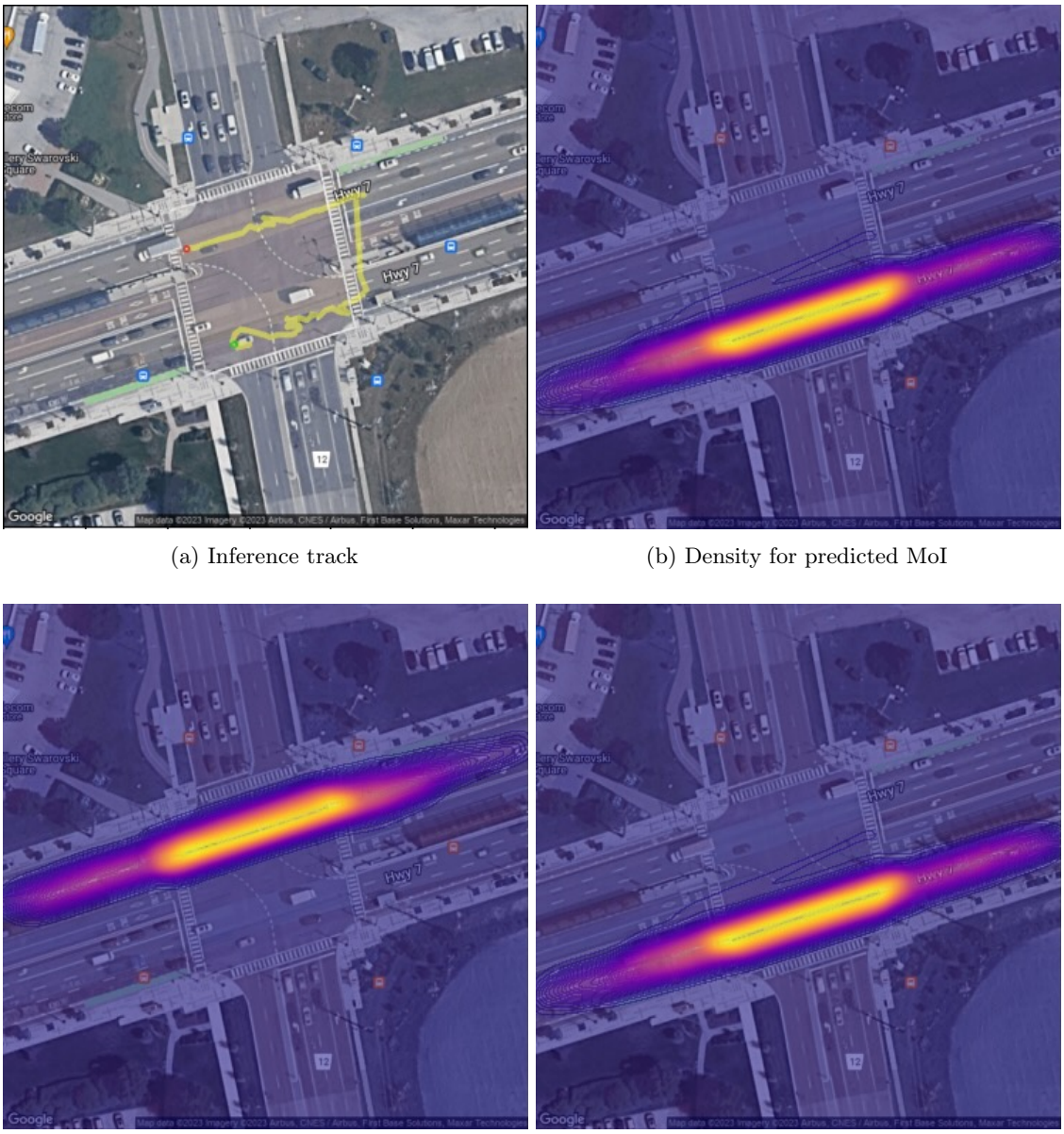
(b) Density for predicted MoI



(c) Density for MoI 2

(d) Density for MoI 7

Figure 39: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 11 (b) while the track, due to re-id error, is a combination of MoI 2 (c) and 7 (d).



(a) Inference track

(b) Density for predicted MoI

(c) Density for MoI 5

(d) Density for MoI 11

Figure 40: An example of a failure mode from our single-camera ground-plane ML counter on the Region of York validation set. The start and end of the inference track (a) are highlighted in green and red respectively. The system incorrectly predicts an MoI of 11 (b) while the track, due to re-id error, is a combination of MoI 5 (c) and 11 (d).

6.5 Conclusion

6.5.1 Superiority of Ground Plane Reasoning for TMC

Our study shows that ground plane reasoning is better than image plane reasoning for TMC. By using real-world back-projected coordinates, we improved TMC error rates.

6.5.2 Superiority of Maximum Likelihood Method

We found that to fully use the information from all points in a track, we need flexible probabilistic models. Simple models like VOTE don't capture important cues and are influenced by inliers. Interestingly, straightforward methods that classify based on the entry and exit points of a track are a good alternative. This indicates that the endpoints of the tracks are more telling of the turn type than the middle points.

6.5.3 Impact of Camera Sway

Camera movement can affect how well our model works. We found that the performance of our best detector (ground-plane ML) was affected by camera sway in the Trans-Plan dataset.

6.5.4 Impact of Fine-Tuning the Detector

For the Region of York dataset, fine-tuning the detector led to an improvement of less than 2%, while for Trans-Plan and CityFlow datasets performance declined. This shows that fine-tuning models on the Region of York dataset worsens generalization. As such, using a pre-trained detector may be a better choice.

6.5.5 Failure Modes

Our analysis indicates that the majority of errors in our system originate from issues in the tracking module, including fragmented and prematurely terminated tracks as well as re-identification errors. Addressing these challenges through improved track reassembly, better handling of video terminations, and enhanced re-identification techniques could significantly boost overall system performance.

6.5.6 Assumptions

Our method assumes we're looking at a flat intersection from a fixed camera with a broad view that captures all corners. This could be generalized to non-flat intersections my making use of readily-available terrain models.

7 Multi-Camera Trajectory Classification

Multi-camera setups may address issues (eg. occlusion) common in single-camera setups. This section explores the potential of multi-camera integration for TMC in the ground-plane reasoning context.

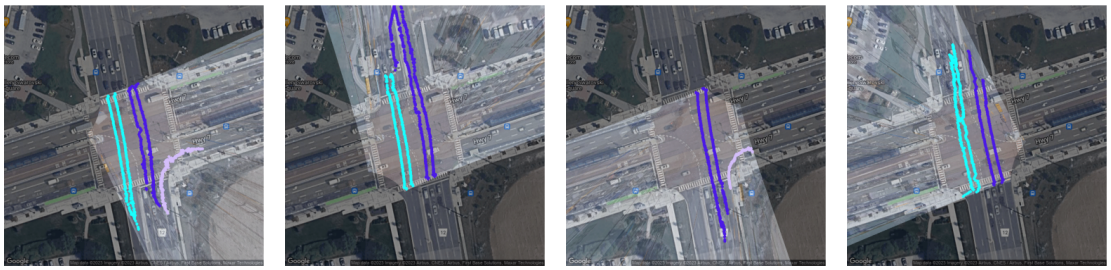
7.1 Background

Prior work [1, 54] has explored back-projection to the ground plane for classification of turning movement counts and integration of multiple camera views, however whether this afford significant advantage over single-camera image-plane classification has not been systematically tested.

7.2 Method

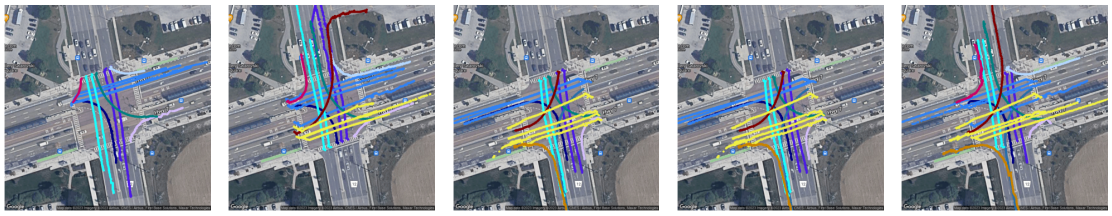
7.2.1 Multi-Camera Prototype Selection

In a multi-camera system, each camera has a different view of the same intersection and may see certain MoIs better than others (Figure 41). When reasoning on the ground plane, due to the shared global coordinate system, training tracks from all cameras can be used for prototype selection, potentially resulting in better prototypes than could be obtained from a single camera (Figure 42).



(a) Camera 1 (b) Camera 2 (c) Camera 3 (d) Camera 4

Figure 41: Prototype tracks captured from each of the four short-range cameras in the Region of York dataset over the same time interval. Each camera captures certain MoIs better than others.



(a) Camera 1 (b) Camera 2 (c) Camera 3 (d) Camera 4 (e) All 4 cameras

Figure 42: Prototype tracks captured with individual and multiple cameras in the Region of York dataset over the entire training period. Note how selecting prototypes using all cameras results in a more complete set of prototypes.

7.2.2 Multi-Camera Integration at the Counting Level

We investigate a simple weak fusion approach to integrate information from multiple cameras. Specifically we assign each of the four cameras to be responsible for only a subset of movements. To derive this assignment we assess the performance of each camera with respect to each MoI individually and assign each movement to the camera with the lowest TMC error. In cases of a tie, we adjust to have roughly the same number of movements assigned to each camera.

7.3 Evaluation

7.3.1 Multi-Camera TMC Evaluation

Multi-camera turn count classification is assessed by the number of count errors, with camera $c(j)$ being responsible for MoI j:

$$MAE = \frac{1}{N}\sum_{j=1}^{12}|\hat{n}_{j,c(j)} - n_j|$$

where $\hat{n}_{j,c(j)}$ is the estimated count for MoI j using camera $c(j)$, n_j is the ground truth count for MoI j, and $N = \sum_{j=1}^{12} n_j$ is the total ground truth count.

7.3.2 Multi-Camera TMC Bias

Multi-camera turn count classification bias is assessed by the difference between the estimated and actual number of movements, with camera $c(j)$ being responsible for MoI j:

$$\text{Bias} = \frac{\hat{N} - N}{N} \tag{30}$$

where $\hat{N} = \sum_{j=1}^{12} \hat{n}_{j,c(j)}$ represents the total estimated counts. We only report TMC bias in Section 7.4.2 to provide deeper insight into how multi-camera integration improves TMC performance.

7.4 Results

7.4.1 Integration at the Counting Level

We evaluate our simple multi-camera integration approach on the Region of York dataset, as it includes multiple cameras, with pre-trained InternImage as detector, ByteTrack as tracker, and ML as counter. The turning movement count error of each camera with respect to each MoI on the training partition is reported in Table 22 - (a) and derived camera-to-MoI assignment is given in Table 22 - (b). This resulted in assigning between 2-4 classes to each camera.

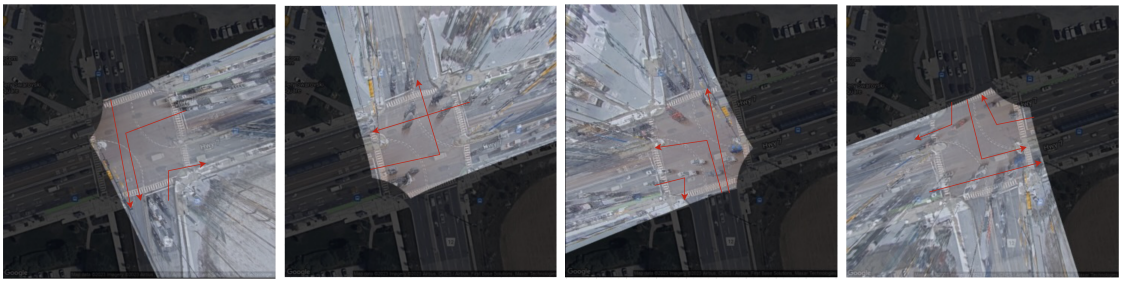
MOI	Cam1	Cam2	Cam3	Cam4
1	9.52%	10.52%	**4.76%**	33.33%
2	5.00%	3.75%	**2.50%**	5.00%
3	**50.00%**	**50.00%**	88.88%	77.77%
4	**0.00%**	**0.00%**	100.00%	400.00%
5	4.23%	**3.19%**	9.57%	24.46%
6	40.00%	40.00%	70.00%	**20.00%**
7	18.51%	7.40%	18.51%	**0.00%**
8	**6.41%**	7.69%	8.97%	7.69%
9	20.00%	28.57%	50.00%	**14.28%**
10	66.66%	**33.33%**	66.66%	100.00%
11	10.72%	5.79%	6.08%	**5.74%**
12	53.84%	63.63%	**38.46%**	76.92%

(a)

MOI	Cam1	Cam2	Cam3	Cam4
1	-	-	✓	-
2	-	-	✓	-
3	✓	-	-	-
4	✓	-	-	-
5	-	✓	-	-
6	-	-	-	✓
7	-	-	-	✓
8	✓	-	-	-
9	-	-	-	✓
10	-	✓	-	-
11	-	-	-	✓
12	-	-	✓	-

(b)

Table 22: (a) TMC error rates on Region of York training data by MoI and camera. (b) MoI to camera assignment.



(a) Camera 1 (b) Camera 2 (c) Camera 3 (d) Camera 4

Figure 43: MoIs assigned to each of the four short-range cameras in the Region of York dataset.

7.4.2 Evaluation of Multi-Camera system

Following the derived assignment in Table 22 - (b), we assess the ML counter in single-camera, multi-camera, and oracle modes on the test partition of the Region of York dataset. In oracle mode, we used ground truth tracks; for both single-camera and multi-camera modes, we used pre-trained InternImage as detector and Bytetrack as tracker. In addition to TMC MAE, we report TMC bias for this experiment in Table 23 to highlight the impact of our multi-camera integration on total detected movements. We found that even weak fusion of the four cameras can substantially reduce error rate as well as bias by 5%. Note that the single-view system tends to over-count, presumably due to fragmentation of tracks caused by occlusions from other vehicles and infrastructure. This reflects the fact that the best vantage depends upon the MoI. Comparing the performance of oracle mode with that of multi-camera mode suggests that errors from the tracking module account for 3.93% (7.39% - 3.46%) MAE and 0.06% bias, while errors in the counting module account for 3.46%

MAE. This indicates that enhancing tracking performance can be as impactful as improving the counting module for the overall system.

Mode	MAE	Bias
Single Camera	13.82%	5.27%
Multi Camera	**7.39%**	**0.06%**
Oracle	**3.46%**	**0.00%**

Table 23: TMC performance (mean absolute error and mean bias over turning movement classes) of single- and multi-camera (weak fusion) systems on the Region of York test partition. Oracle result comes from ground truth detection and tracking with our multi-camera system.

7.5 Conclusion

7.5.1 Multi-Camera integration for TMC

Ground plane reasoning works well in environments that use multiple sensors. We found considerable improvement in TMC error rates even with a simple weak fusion method that assigns each camera to specific MOIs.

8 Future Work

Our TMC pipeline incorporates some manual steps including annotating key points used to estimate the homography and defining the intersection ROI. It would be useful to explore ways to fully automate these steps.

Currently, InternImage, a large foundational model, isn't fast enough for real-time use. We should look into ways of using a real-time detector that will still result in good TMC performance.

We have only tested trackers that operate in real-time. It would be interesting to see how offline trackers, which consider global temporal information, might improve performance.

Our method for mapping objects from camera view to the ground plane assumes a flat surface at the intersection. Using terrain models could remove this limitation.

We currently estimate the contact point of vehicles with the road using the bottom center of a bounding box, which isn't very accurate. Segmentation masks or monocular 3D detection, may be used to improve accuracy.

Our weak fusion method for multi-camera integration is primitive. A more thorough integration across all parts of the system — detection, tracking, and counting — could be beneficial.

Our datasets only provide course-grained turning movement counts by providing total number of counts for each MoI. Analyzing TMC with fine-grained labels that assign an MoI label to each track could offer deeper insights.

References

[1] A. Abdelhalim and M. Abbas. Towards real-time traffic movement count and trajectory reconstruction using virtual traffic lanes. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2527–2533, 2020.

[2] L. R. Agostinho, N. M. Ricardo, M. I. Pereira, A. Hiolle, and A. M. Pinto. A practical survey on visual odometry for autonomous driving in challenging scenarios and conditions. *IEEE Access*, 10:72182–72205, 2022.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] A. Badithela, T. Wongpiromsarn, and R. M. Murray. Evaluation metrics for object detection for autonomous systems. *arXiv preprint arXiv:2210.10298*, 2022.

[5] P. Barcellos, C. Bouvié, F. L. Escouto, and J. Scharcanski. A novel video based system for detecting and counting vehicles at user-defined virtual loops. *Expert Systems with Applications*, 42(4):1845–1856, 2015.

[6] E. Bas, A. M. Tekalp, and F. S. Salman. Automatic vehicle counting from video for traffic flow analysis. In *2007 IEEE Intelligent Vehicles Symposium*, pages 392–397, 2007.

[7] M. Bashar, S. Islam, K. K. Hussain, M. B. Hasan, A. B. M. A. Rahman, and M. H. Kabir. Multiple object tracking in recent times: A literature review. *arXiv preprint arXiv:2209.04796*, 2022.

[8] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008(1):246309, 2008.

[9] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, September 2016.

[10] A. Bochkovskiy, C. Wang, and H. M. Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[11] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[12] K. N. Bui, H. Yi, and J. Cho. A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2466–2474, 2020.

[13] O. Cetintas, G. Brasó, and L. Leal-Taixé. Unifying short and long-term tracking with graph hierarchies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22877–22887, 2023.

[14] M. Chang, C. Chiang, C. Tsai, Y. Chang, H. Chiang, Y. Wang, S. Chang, Y. Li, M. Tsai, and H. Tseng. AI city challenge 2020 – computer vision for smart transportation applications. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2638–2647, 2020.

[15] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. MMDetection: Open MMLab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[16] C. Creß, W. Zimmer, L. Strand, M. Fortkord, S. Dai, V. Lakshminarasimhan, and A. Knoll. A9-dataset: Multi-sensor infrastructure-based dataset for mobility research. In *IEEE Intelligent Vehicles Symposium (IV)*, 2022.

[17] P. Devadhas Sujakumari and P. Dassan. Generative adversarial networks (GAN) and hdfs-based realtime traffic forecasting system using CCTV surveillance. *Symmetry*, 15(4), 2023.

[18] A. Dore, A. Beoldo, and C. S. Regazzoni. Multitarget tracking with a corner-based particle filter. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 1251–1258, 2009.

[19] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. CenterNet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[20] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[21] M. Fitzpatrick. *Create GUI Applications with Python and Qt5 (PyQt5 Edition): The hands-on guide to making apps with Python*. M. Fitzpatrick, 2020.

[22] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. YOLOX: Exceeding YOLO series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[23] D. Gloudemans and D. B. Work. Fast vehicle turning-movement counting using localization-based tracking. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4150–4159, 2021.

[24] S. Godsill. Particle filtering: The first 25 years and beyond. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7760–7764, 2019.

[25] S. V. Ha, N. M. Chung, T. Nguyen, and H. N. Phan. Tiny-pirate: A tiny model with parallelized intelligence for real-time analysis as a traffic counter. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4114–4123, 2021.

[26] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[28] J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[30] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023.

[31] The MathWorks Inc. Optimization toolbox version: 9.4 (r2022b), 2022.

[32] Itseez. *The OpenCV Reference Manual*, 2.4.9.0 edition, April 2014.

[33] H. Izadinia, I. Saleemi, W. Li, and M. Shah. MP2T: Multiple people multiple parts tracker. In A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, editors, *Computer Vision – ECCV 2012*, pages 100–114, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[34] Z. Karwand, S. Mokhtar, K. Suzuki, O. Oloruntobi, M. Z. Shah, and S. H. Misnan. Impact of splitter-island on pedestrian safety at roundabout using surrogate safety measures: A comparative study. *Sustainability*, 15(6), 2023.

[35] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[36] Z. Kim. Real time object tracking based on dynamic feature grouping with background subtraction. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[37] V. Kocur and M. Ftáčnik. Multi-class multi-movement vehicle counting based on centertrack. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4004–4010, 2021.

[38] J. Kotzenmacher, E. D. Minge, and B. Hao. Evaluation of portable non-intrusive traffic detection system. *IMSA Journal*, 43(3), 2005.

[39] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

[40] Q. Li, R. Li, K. Ji, and W. Dai. Kalman filter and its application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pages 74–77, 2015.

[41] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018.

[42] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *The European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[43] Z. Liu, W. Zhang, X. Gao, H. Meng, X. Tan, X. Zhu, Z. Xue, X. Ye, H. Zhang, S. Wen, and E. Ding. Robust movement-specific vehicle counting at crowded intersections. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2617–2625, 2020.

[44] J. Lu, M. Xia, X. Gao, X. Yang, T. Tao, H. Meng, W. Zhang, X. Tan, Y. Shi, G. Li, and E. Ding. Robust and online vehicle counting at crowded intersections. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3997–4003, 2021.

[45] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T. Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[46] C. Ma, L. Qiao, C. Zhu, K. Liu, Z. Kong, Q. Li, X. Zhou, Y. Kan, and W. Wu. Holovic: Large-scale dataset and benchmark for multi-sensor holographic intersection and vehicle-infrastructure cooperative. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22129–22138, Los Alamitos, CA, USA, Jun 2024. IEEE Computer Society.

[47] T. Meinhardt, A. Kirillov, L. Leal-Taixé, and C. Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8834–8844, 2022.

[48] P.G. Michalopoulos. Vehicle detection video through image processing: the autoscope system. *IEEE Transactions on Vehicular Technology*, 40(1):21–29, 1991.

[49] D. Mitzel, E. Horbert, A. Ess, and B. Leibe. Multi-person tracking with sparse detection and continuous segmentation. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 397–410, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[50] D. Mitzel and B. Leibe. Real-time multi-person tracking with detector assisted structure propagation. In *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pages 974–981, 2011.

[51] M. L. Monteiro, A. Bernardino, and H. S. Pinto. Bytecounter: Improving vehicle turning-movement counting. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3456–3462, 2023.

[52] W. Mou, H. Wang, G. Seet, and L. Zhou. Robust homography estimation based on non-linear least squares optimization. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 372–377, 2013.

[53] V. Movahedi and J. H. Elder. Design and perceptual validation of performance measures for salient object segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 49–56, 2010.

[54] K. Nakano, M. Nakazawa, and M. Zuzak. Complementing vehicle trajectories using two camera viewpoints. In *2024 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6, 2024.

[55] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M. Chang, X. Yang, Y. Yao, L. Zheng, P. Chakraborty, C. E. Lopez, A. Sharma, Q. Feng, V. Ablavsky, and S. Sclaroff. The 5th AI city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021.

[56] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M. Chang, X. Yang, L. Zheng, A. Sharma, R. Chellappa, and P. Chakraborty. The 4th AI City Challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2665–2674, June 2020.

[57] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M. Chang, Y. Yao, L. Zheng, M. Shaiqur Rahman, A. Venkatachalapathy, A. Sharma, Q. Feng, V. Ablavsky, S. Sclaroff, P. Chakraborty, A. Li, S. Li, and R. Chellappa. The 6th AI City Challenge. In *2022 IEEE/CVF Conference*

on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 3346–3355. IEEE Computer Society, June 2022.

[58] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *2016 European Conference on Computer Vision (ECCV)*, pages 483–499, 2016.

[59] N. Nigam, D. P. Singh, and J. Choudhary. A review of different components of the intelligent traffic management system (itms). *Symmetry*, 15(3), 2023.

[60] A. Ospina and F. Torres. Countor: count without bells and whistles. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2559–2565, 2020.

[61] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on lie algebra. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 728–735, 2006.

[62] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[63] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision (ECCV)*, pages 17–35. Springer, 2016.

[64] M. S. Shirazi and B. T. Morris. Vision-based turning movement monitoring:count, speed & waiting time estimation. *IEEE Intelligent Transportation Systems Magazine*, 8(1):23–34, 2016.

[65] M. S. Shirazi and B. T. Morris. Trajectory prediction of vehicles turning at intersections using deep neural networks. *Machine Vision and Applications*, 30(6):1097–1109, 2019.

[66] M. S. Shirazi and A. Patooghy. A vision-based framework for intersection monitoring and signal evaluation. In *2021 IEEE International Smart Cities Conference (ISC2)*, pages 1–4, 2021.

[67] X. Song and R. Nevatia. Detection and tracking of moving vehicles in crowded scenes. In *2007 IEEE Workshop on Motion and Video Computing (WMVC'07)*, pages 4–4, 2007.

[68] Sukriti Subedi and Hua Tang. Development of a multiple-camera 3d vehicle tracking system for traffic data collection at intersections. *IET Intelligent Transport Systems*, 13(4):614–621, 2019.

[69] D. Sugimura, K. M. Kitani, T. Okabe, Y. Sato, and A. Sugimoto. Using individuality to track individuals: Clustering individual trajectories in crowds using local appearance and frequency trait. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1467–1474, 2009.

[70] Z. Tang, M. Naphade, M. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J. Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8797–8806, Long Beach, CA, USA, 2019.

[71] Torben Teepe, Philipp Wolters, Johannes Gilg, Fabian Herzog, and Gerhard Rigoll. Lifting multi-view detection and tracking to the bird's eye view. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 667–676, 2024.

[72] D. N. Tran, L. H. Pham, H. Nguyen, T. H. Tran, H. Jeon, and J. W. Jeon. A region-and-trajectory movement matching for multiple turn-counts at road intersection on edge device. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4082–4089, 2021.

[73] Arpitsinh Vaghela, Duo Lu, Aayush Atul Verma, Bharatesh Chakravarthi, Hua Wei, and Yezhou Yang. Mc-bevro: Multi-camera bird eye view road occupancy detection for traffic monitoring. *arXiv preprint arXiv:2502.11287*, 2025.

[74] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[75] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14408–14419, 2023.

[76] W. Wang, T. Gee, J. Price, and H. Qi. Real time multi-vehicle tracking and counting at intersections from a fisheye camera. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 17–24, 2015.

[77] Z. Wang, B. Bai, Y. Xie, T. Xing, B. Zhong, Q. Zhou, Y. Meng, B. Xu, Z. Song, P. Xu, R. Hu, and H. Chai. Robust and fast vehicle turn-counts at intersections via an integrated solution from detection, tracking and trajectory modeling. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2598–2606, 2020.

[78] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu. UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 193:102907, 2020.

[79] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017.

[80] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL programming guide: The official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[81] Y. Xia, X. Shi, G. Song, Q. Geng, and Y. Liu. Towards improving quality of video-based vehicle counting method for traffic flow estimation. *Signal Processing*, 120:672–681, 2016.

[82] M. Yokoyama and T. Poggio. A contour-based moving object detection and tracking. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 271–276, 2005.

[83] H. Yu, Y. Luo, M. Shu, Y. Huo, Z. Yang, Y. Shi, Z. Guo, H. Li, X. Hu, J. Yuan, et al. DAIR-V2X: A large-scale dataset for vehicle-infrastructure cooperative 3D object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21361–21370, 2022.

[84] L. Yu, Q. Feng, Y. Qian, W. Liu, and A. G. Hauptmann. Zero-virus: Zero-shot vehicle route understanding system for intelligent transportation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2534–2543, 2020.

[85] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[86] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European Conference on Computer Vision*, pages 1–21. Springer, 2022.

[87] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[88] M. Zhong and G. Liu. Establishing and managing jurisdiction-wide traffic monitoring systems: North American experiences. *Journal of Transportation Systems Engineering and Information Technology*, 7(6):25–38, 2007.