INTERSTATE 24 MOTION: A MULTI-CAMERA VEHICLE TRACKING INSTRUMENT FOR LARGE-SCALE TRAJECTORY EXTRACTION

By

Derek Adam Gloudemans

Dissertation
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in

Computer Science

December 16, 2023

Nashville, Tennessee

Approved

Daniel B. Work (Chair), Ph.D.

Douglas Adams, Ph.D.

Alexandre Bayen, Ph.D.

Gautam Biswas, Ph.D.

Forrest Laine, Ph.D.

This dissertation is dedicated to Aaron Swartz and Alexandra Elbakyan, and to everyone else fighting to make science worth contributing to. Thank you for believing in a world that I want to be a part of.

ACKNOWLEDGMENTS

This dissertation is the sum total of many ideas, collaborations, years of work, conversations, kind gestures, grant proposals, leaps of faith, drives to the crag down I-24, rounds of volleyball, porch chats, rainy days, late nights coding, runs through Shelby Bottoms, and many many other innumerable pieces. I can only hope to acknowledge a small fraction of my support here.

My advisor: First and foremost, I owe a tremendous debt of gratitude to my advisor Dr. Dan Work. You have been at once the most positive, supportive, and level-headed advisor I could ever have asked for, as well as the most ambitious. 5 years ago, I had no particular passion for work on traffic, but I had a particular passion for problems so difficult they might not be possible. On this project we've witnessed improbable victories, crushing setbacks, hair-pulling technical dilemmas, and monotonous, sleep-deprived nights of work, and throughout all of that you have charted an optimistic, bold, and clear path of progress. I cannot thank you enough for your endless patience and support. I am, at my heart, a person who loves to build things; what a joy it has been to work with someone who has ideas worth building!

My research community: To my committee, thank you for your valuable time, support, and feedback, and for taking an interest in my work. To my lab-mates, for making every day in the lab, without exception, a little bit brighter, thank you. I have enjoyed collaborating with you and being challenged by you. Yanbing, sorry for every algorithmic failure I created that you had to solve downstream. Gergely, sorry for every "great idea" I had that meant another 10 hours of work for you. Y'all are truly the best coworkers I could ask for. To the CIRCLES research group, it has been a great joy to participate in projects far larger and bolder than I could have accomplished on my own. Thank you for dreaming big. To the Vanderbilt Engineering community, especially Dr. Troxel, Dr. Duddu, Phil Davis, Doc Bob, Rich Teising and Dr Lin, thank you for believing in me and/or letting me use your welding machine after-hours. And lastly, to everyone else who poured countless hours into making it possible, especially Brad Freeze, Said ElSaid, Meredith Cebelak, Matt D'Angelo, Lee Smith, Craig Philip, Eric Hall, and Janos Sztipanovits, thank you for believing in this project long enough to get it off the ground.

My family: Each one of you has inspired me in ways I can only hope to imitate. I am lucky to count myself among a slim majority of PhD students who have co-authored a paper with my sister (thanks for working with me Nikki), and an even slimmer subset who have conducted a large-scale traffic experiment with my parents, easily the best experience of my 5-year stint. You are the best parents I could possibly have asked for. And to Grandpa Gloudemans, thank you for laughing when I sent boards flying off the table saw and through your garage door, and for giving me the confidence to be truly curious and to make some mistakes for myself. I love y'all and owe pretty much everything I am to you!

My friends: To my climbing friends, thank you for catching me on my largest falls and for keeping me on the line when I wanted to call it quits. I wouldn't have finished this work without our mountain and desert adventures. To my volleyball friends, thank you for making every week fun and for bearing with my grumbling. I owe you many perfect sets. To Corbin, thanks for an endless stream of ideas and for putting up with me these past 5 years. I've never met a person who was more fun to look at the world with.

Funders: Finally, I have been generously supported throughout my work by the National Science Foundation's Graduate Research Fellowships Program (Grant DGE-1937963), by the Federal Highway Administration's Dwight David Eisenhower Transportation Fellowship Program (Grant 693JJ32245006), and by Vanderbilt University's Russell G. Hamilton Scholar Program. I am tremendously grateful to have been chosen for these programs from a wide pool of equally qualified peers. I am also grateful to have been funded in part by grants from the Department of Energy (EERE Grant CID DE-EE0008872), the Tennessee Department of Transportation (CMAQ Grant TN20210003), and the National Science Foundation (Grant 2135579). The views expressed herein do not necessarily represent the views of the Tennessee Department of Transportation, U.S. Department of Energy, or the United States Government.

Everyone else: If you've made it this far and haven't seen yourself mentioned, this one is for you. I can't imagine what this journey would have looked like without the little bits of support from everyone I met along the way.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Foreword

You, regrettably, are a bad driver. I asked your friends and they told me the cold, harsh truth. But more importantly, (and more accurately), data can prove it. And in fact, as traffic scientists will delight to inform you, the only "good" drivers around are, in fact, traffic scientists! Or more precisely, the vehicle controllers designed by traffic scientists.

Now before I risk alienating possibly one of the only 10 people who will ever lay eyes on this dissertation, let's take a moment to dissect what is meant by "good" and "bad" drivers. *Traffic* is a set of complex phenomena caused by the interaction of numerous individual drivers at scale. That is, individual behaviors can result in emergent phenomena visible in macro-scale properties of traffic such as vehicular density, flow, and speed through a section of roadway. One such traffic phenomenon is *traffic hysteresis*, or "stop and go waves", which result when small perturbations in driver behaviors are amplified cause rippling, amplifying downstream effects [7, 8]. Recent years have brought on a large host of research utilizing *intelligent transportation systems* (ITS) built on new technologies to mitigate these negative emergent phenomena, increasing safety and mobility. For instance, it has been shown that a modest number of well-designed autonomous vehicles ("good" drivers) can effectively dampen stop-and-go waves caused by "bad" human drivers [9], reducing congestion and drastically reducing overall energy usage on the roadway [10]. Studies in a similar vein [11] have showed that *platooning* or grouping trucks together on highways has a positive influence on traffic flow near merging and diverging areas, and [12, 13] demonstrated that energy-saving control strategies could be deployed on commercially available vehicles with little additional hardware needed. By comparison to these carefully designed control algorithms, most human drivers are "bad" in terms of their effects on the flow of traffic around them (at least in some traffic regimes), A well-informed driver could approximate or even outperform one of these isolated control strategies, but newer generations of *connected autonomous vehicle* control strategies [14, 15] share information across vehicles throughout a traffic flow, providing valuable information for control and decision-making that even the best human drivers do not have access to.

On the other hand, the simple fact that ITS is used does not a good driver make. Many existing adaptive cruise controllers themselves contribute to amplifying traffic waves [16], and the same study on truck platooning above [11] also found that truck platoons made it more difficult for merging vehicles enter the flow of traffic. Great care is required when selecting algorithms to deploy on real roadways, especially when they will be deployed across tens of thousands of vehicles. How are we to know which control algorithms benefit traffic as a whole, and which strategies are having unintended repercussions? What's a traffic control designer to do?!

I'm glad you asked![1] By recording precise positional data for all vehicles in a traffic flow at fine-grained intervals (called *trajectory data*), researchers can analyze how various deployed technologies affect the flow of traffic around them, both at a local scale (e.g. an aggressive merge strategy caused the following driver to undergo an unsafe deceleration to avoid a collision) and at a global scale (the deceleration caused a wave of deceleration to propagate backwards from the initial incident for half a mile). This allows them to separate the good (stabilizing, flow-increasing) controllers from the bad (unsafe, flow-reducing, wave-causing). Historically, this data has been extremely difficult to obtain, as it requires observing every vehicle (which one cannot hope to accomplish with GPS sensors alone) at fine-grained time intervals and over long distances (individual fixed sensors such as cameras or radar units do not provide enough coverage).

This thesis presents a new instrument for obtaining vehicle trajectory data at a scale previously impossible. Interstate-24 MOTION is a 4.2 mile stretch of roadway near Nashville, Tennessee, densely instrumented such that every vehicle's position is precisely recorded for the entire roadway segment. This instrument enables a new paradigm of traffic research where researchers can develop a ITS technologies to benefit traffic and then deploy them on a real roadway to gather empirical data supporting or disproving their theoretical claim. Who knows, one day data from Interstate-24 MOTION may even validate your status as a "good" driver! (Just kidding, I-24 MOTION trajectories are anonymous, so we'd never know it was you). In any case, this new observational technology offers new abilities to design the next generation of vehicular autonomy; control algorithms that not only move people from point A to point B, but also improve safety, fuel efficiency, and even travel times for the entire flow of traffic along the way.

[1] implicitly, by not skipping the foreword.

1. Introduction

Nearly 100 years ago, Bruce Greenshields strode into a grassy field along US Highway 23 near Deleware, Ohio, armed with a video camera. Greenshields sought to develop models for understanding traffic based on empirical data, inspiring a flourishing field of *macroscopic* traffic research in his wake. At the macroscopic level, traffic phenomena are often observed and described with three quantities of interest, i.e., flow, speed, and density [17]. Fundamental diagrams [18] like the Greenshields and Greenberg models [19, 20] relate the traffic quantities while models such as the Lighthill-Whitham-Richards (LWR) [21] and the Aw–Rascle–Zhang (ARZ) [22] are developed to describe the spatio-temporal evolution of traffic. Leveraging state of the art technologies of the day, researchers worked to validate these models with data collected from radar-based devices and loop detectors [23]. Large-scale macroscopic data monitoring systems such as the freeway performance measurement system (PeMS) [24] in the United States; the A5 freeway near Frankfurt [25] in Germany; and the M42 highway [26] in England; and later floating-vehicle measurement-based on cell phone carrier data [27] or GNSS positional data [28] have enabled research on macroscopic traffic flow dynamics [29–33]. A challenge is that the data typically must be interpolated spatially (in the case of inductive loops), or scaled up across all vehicles (in the case of probe data) to gain a complete spatio-temporal picture [1]

Unlike the accumulated average macroscopic data and models, *microscopic* traffic models give attention to the interactions between individual vehicles. Since the early car-following experiments [35] conducted by physically connecting vehicles to measure space gap, many emerging in-vehicle technologies including on-board radar detectors [36], cameras [37], laser sensors [38] and GNSS devices [39, 40] have been applied to measure vehicle spacing, speed and relative speed.

Some traffic phenomena benefit from observation of traffic across the micro and macroscopic scales. For example, traffic waves observable at the macroscopic scale can result from instabilities and disturbances in the flow at the level of individual vehicles [7, 8]. Macroscopic data, frequently used for traffic wave studies, can cover a great spatiotemporal scale that reveals the dynamics of traffic waves on road networks, but it is unable to provide insight into why the wave is generated and how it is propagated. Macro-microscopic *vehicle trajectory data* can help provide these insights [33, 41, 42] by providing precise positional and derivative data *for every vehicle in a traffic flow*, when available with adequate spatio-temporal coverage. Trajectory data with the complete information for specific road segments supported a range of efforts including the development, calibration and validation of car-following models [43, 44], lane-change modeling, trajectory prediction [45, 46], and traffic oscillation analysis [47]. Figure 1.1

However, historically this vehicle trajectory data data was tremendously labor-intensive to collect, requiring manual annotation of vehicle positions in recorded videos of traffic [8], approximation with spare

[1]Parts of this introduction are adapted from [34].

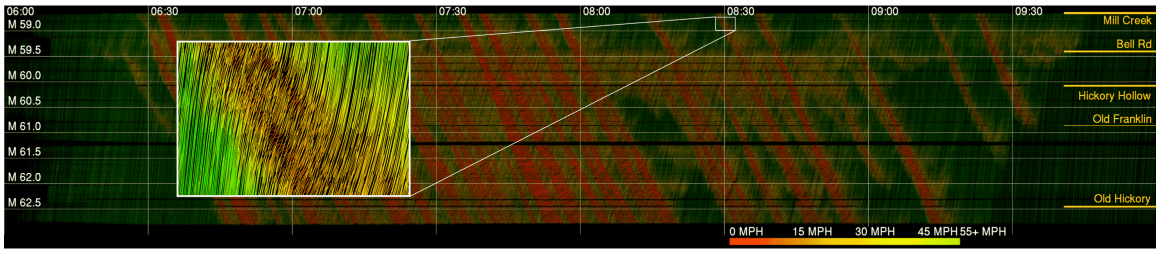


Figure 1.1: Example vehicle trajectory data produced in this work. Each vehicle is plotted as a line colored according to its instantaneous speed. x-axis: time of day (HH:MM); y-axis: roadway postmile (mi). Postmile decreases for travelers in the westbound direction. A typical congestion pattern is shown with frequent oscillatory traffic observed; and recurring waves travel upstream relative to the direction of traffic at 12-13 mph.The figure inset shows a zoomed in portion of the data which is 0.25 mi in length and 4 min in duration. (Credit: Gergely Zachár and Derek Gloudemans.)

instrumented vehicles [48], or semi-automatic analysis and subsequent manual correction [49, 50]. As a result, there are still many open questions on the nature of the traffic wave, such as the general growth and propagation pattern of traffic wave [42, 51, 52], and the relationship between individual instability and the macroscopic traffic wave [53]. Hence, abundant trajectory datasets, as highlighted in the article [54], can enable traffic research at both the macroscopic and microscopic scales, aiding in understanding traffic phenomena like jam clusters and state transition dynamics [25, 31, 55]. It can also capture the complex interaction within multiple-class traffic participants for heterogeneous traffic flow [56–58].

A few recent trends have lowered the barrier to creating new trajectory data. First, significant research has been devoted to the computer vision tasks of *object detection* (locating relevant objects within an image) and *object tracking* (associating distinct objects in video frames across time). Especially in the past 10 years, rapid progress has been made in the use of modern hardware [59], neural network architectures [60–63], and massive-scale image datasets [64, 65] to fit accurate object detection algorithms. Second, HD traffic cameras and aerial drones have become increasingly prevalent. Equipped with these technologies, recent research efforts have revisited the task of vehicle trajectory extraction and made marked advancements to the state of the art. The HighD, [66], ExiD [67], AUTOMATUM [68], and HIGH-SIM [69] datasets all utilize aerial imagery shot from either drone or helicopter-mounted cameras to produce complete highway vehicle trajectory data semi-automatically, and the *Third Generation Simulation* (TGSIM) [70] is a similar in-progress effort designed to capture trajectory data containing deployed automated vehicle technologies. Similarly, the pNEUMA [71], inD [72], rounD [73], OpenDD [74], Interaction [75] and CitySim [76] datasets utilize drones or swarms of drones to study complex urban vehicle and pedestrian interactions in more detail. High aerial fields of view make modern image segmentation algorithms [77] well posed for vehicle tracking in these contexts. Unfortunately, these methods are temporally limited by the relatively short battery life of drones (generally under an hour) and the requirement for human pilots, and spatially limited by the relatively small field of view of a single drone.

A different vein of research aims to revisit the approach of NGSIM with updated technology, utilizing fixed cameras (or other sensors) to produce vehicle trajectories for longer durations. Work on the Minnesota Traffic Observatory [78] and more recently on the Lower Saxony Testbed [79] and Zen Traffic Roadways [80] follow this approach. Even so, fixed infrastructure deployment have been stymied by the inherent challenges of vehicle trajectory observation in this regime, namely: i.) vehicles can be significantly occluded by taller vehicles and by roadway infrastructure ii.) there are few datasets on which to train precise 3D vehicle tracking algorithms, and iii.) high performing tracking and detection methods still run below 30 frames per second on a GPU for frames of modest size (e.g., 960×540 [1], 1392×512 [81], and 1920×1080 [82]. Thus, a continuous source of publicly available trajectory data, to the best of our knowledge, does not yet exist.

Today a confluence of developments promises to bring about paradigm-shift transportation science. On the one hand, new technologies within the past decade have enabled trajectory data gathering at scales that were previously infeasible. On the other hand, increasingly automated vehicles are being developed and deployed on roadways, changing the fundamental physics of traffic flow. Even a small number of automated vehicles can have a direct impact on the macroscopic behavior of traffic flow [10, 83]. Thus, there is at once a capability and a great need to monitor and observe traffic flows across microscopic and macroscopic scales.

This dissertation sits in the space created by this technological confluence, answering the following question: *How can we create a large-scale traffic observation instrument for accurate and persistent vehicle trajectory generation using computer vision techniques?* Work to answer this question falls into three major categories. First, a cutting-edge traffic instrument for rich data collection is proposed, designed, developed, and implemented. To extend the state of the art trajectory datasets which are limited on-off efforts with limited temporal and spatial scope, this work requires an infrastructure sensor-based approach. This in turn requires aggregating vehicle trajectories across a large set of sensors to allow for suitably dense coverage over a large spatial range, and introduces the challenge of vehicle occlusion by other vehicles and by overpasses. Second, computationally efficient algorithms for object detection and tracking are developed that utilize context from the object tracking problem to address these tasks jointly. Fast object tracking is continued challenge within the field because conventional approaches are computationally intensive and are ill-suited to leverage object priors or object patterns within a scene to speed object detection and tracking. This method is then further extended by incorporating information from multiple cameras to and the traffic monitoring domain, yielding a state-of-the-art and production-ready algorithm for trajectory generation. The multi-camera nature of this task is difficult because of the need for precise positional information of cameras with respect to the roadway in the face of non-trivial camera motion over time. Third, several first-in-kind datasets are released. A novel dataset enabling vehicle tracking in 3D space across a multiple-camera network is produced, allowing evaluative benchmarking and improvements of existing algorithms and development of new ones for this task. Each of these datasets must recon with the tremendous manual cost of producing perfect ground truth vehicle annotations for a large set of vehicles, over long distances, and at fine-grained time intervals. Additionally, a vehicle trajectory dataset of massive scale is released as the end data product of the proposed instrument.

1.1 Contributions

The primary contribution of this dissertation is the development of a physical traffic observation instrument and accompany software system that enable the persistent collection of vehicle trajectory data along a 4-mile stretch of 4-lane interstate roadway. The work enabling this is divided into three categories:

- **A Large-Scale Traffic Instrument for Trajectory Data Collection**
  Motivated by the above needs, this dissertation proposes, designs, tests and implements the *I-24 Mobility Technology Interstate Observation Network* (MOTION), a densely instrumented freeway that enables continuous, ongoing coverage of a roadway at the fine-grained vehicle trajectory level. The instrument is also the product of continued collaboration with the Tennessee Department of Transportation and industry partners. MOTION consists of a network of 296 traffic pole-mounted 4K resolution cameras recording video data over a 4.2-mile stretch of freeway in its entirety. The raw video data stream exceeds 24 TB/day of traffic data footage that must be processed in real-time to extract precise vehicle locations, trajectories, and other relevant information from the entire monitored portion of roadway. The design of such a large-scale instrument necessarily requires integration of available state-of-the-art algorithm capabilities into every part of the design process. This work discusses motivating design considerations, proposes a cost-effective physical sensor configuration to enable the data extraction requirements, presents preliminary experiments assessing the feasibility of the system, conducted as part of the first phase of the MOTION deployment, and provides a reference for the I-24 MOTION system as built. The main result of this work is the operational traffic instrument itself, and details on this system are published in a conference workshop paper [84], a conference paper [85], a Concept of Operations document submitted to the Tennessee Department of Transportation detailing the goals, requirements, and design considerations of this instrument, and finally a journal paper detailing the full system as initially completed in November 2022 [34].
- **Algorithms**
  - **Multiple Object Tracking.** A novel multiple object tracking method is proposed, utilizing the core intuition that predictable object motion yields strong object location priors *before* objects are detected at a given time. Prior works have utilized this tracking-contextual information to improve the accuracy of MOT methods [86–90]. Conversely, the proposed approach leverages the strong prior intuition to reduce the required computation to detect and track objects by cropping small image portions known to contain object priors and ignoring the rest of each frame. The MOT task is reformulated as a parallel single object tracking task, and a framework is provided to convert an arbitrary existing object detector and online multiple object tracking method easily into a single object tracker to solve this parallel task. Experimental results show i.) the crop-based method increases both the speed and accuracy of an existing object tracker, ii.) the method achieves a new state of the art on the preeminent vehicle tracking benchmark [1], and iii.) the method yields a best result of 150% speedup with no decrease in accuracy. The main result of this work is published in a conference paper [91].
  - **Vehicle Turning Movement Counting.** Based on the crop-based tracking method proposed, a novel intersection turning movement counting algorithm is proposed. Relative to the previous work, the primary contribution of this work is to introduce a new method for object initialization. The previous work relied on periodic detection of full frames to initialize new objects. In this work, leveraging the turning-movement problem context where only objects that pass through certain ”source” and ”sink” regions of the image are relevant for movement counting, source region initialization is proposed. Each manually identified source region is cropped at each frame and also processed by the same object detector as cropped objects, allowing for the detection of

new objects without ever performing object detection on a whole frame. Experimentally, this method achieves competitive performance against other turning movement counting algorithms, and moreover increases speed by 57% relative to an otherwise identical method instead relying on full frame detections. The main result of this work is published in a CVPR workshop paper [92].

  - **Multiple Camera 3D Tracking.** The crop-based single camera tracking is extended to solve the multiple camera tracking problem. Relative to the previous work, the primary contribution of this work is to further reduce the image area that needs to be processed at each frame by leveraging the redundancy of overlapping camera views. This work proposes to score each camera view of each object, and query only the camera with the best view of each object. Secondly, this work proposes a unified curvilinear coordinate system closely fit to the roadway lane markings capable of providing vehicle positional accuracy on the order of 2 feet across cameras while simultaneously aligning vehicle roadway motion along the primary coordinate system axis. These contributions are included in the supplement of a submitted conference paper [93]. Thirdly, this work proposes a new IOU-based loss formulation for 3D object detection from arbitrary viewpoints. Preliminary results show state of the art performance on relevant multiple-camera 3D datasets, and show that the new loss formulation is able to better and faster converge to the optimal parameter values. This work is submitted as a conference paper [94].

- **Datasets**
  - **Multiple Object Tracking Dataset.** This work proposes a new dataset that provides multiple object tracking labels in a shared 3D space, across multiple camera fields of view. Relative to existing works, this dataset is the first to enable the development of 3D vehicle tracking methods in a traffic monitoring context. Moreover, the dataset has more synchronized camera views, more 3D vehicle bounding boxes, and more annotated frames than any other multiple camera multiple object tracking or traffic monitoring dataset. The dataset furthermore combines the richest attributes of each existing class of datasets, enabling research into new tracking tasks. Subsequent analysis is performed to identify primary sources of error in the dataset arising from the difficult of tight spatio-temporal camera synchronization. This work proposes initial solutions to address these sources of error and refine annotated labels to improve trajectory attributes without sacrificing in annotation quality, moving closer to the goal of a fully unified trajectory and image space annotation set. Finally, this work benchmarks a number of existing multi-camera multiple-object tracking approaches on the dataset, showing the implemented methods fall short of desirable performance especially in occluded and dense scenes. This dataset is necessary for the development of fast and accurate multiple-camera 3D vehicle tracking methods as described above. The main result of this work is published in a conference paper [95].
  - **Large Scale Vehicle Tracking Dataset.** This work proposes a new dataset designed to allow the benchmarking of object tracking algorithms for extremely long term tracking performance. It consists of a single scene of video data, 1 hour in duration, simultaneously recorded from 234 overlapping cameras covering 4.2 miles of interstate roadway. A set 270 GPS trajectories recorded over 100 instrumented vehicles on the roadway during the recording duration is manually corrected to ensure positional accuracy. The annotated trajectories persist for an average of 6.6 minutes (11880 frames average at 30 *frames per second* (FPS)) and in general the scene has high object density ($>$500 object typically visible across the scene). This annotation set is suitable for assessing object tracking algorithms along recall-oriented metrics. Initial experiments show that existing high-performing trackers fall well short of acceptable tracking performance on data of this scale, and further work is needed to develop suitable algorithms for long-term tracking tasks. Moreover, we take considerable care to make the data useful for computer vision applications, developing new techniques for keeping camera homographies more accurately aligned than existing stabilization methods allow. This work is submitted as a conference paper [93].
  - **Vehicle Trajectory Dataset.** 2 weeks of vehicle trajectory data are released with the original I-24 MOTION system paper [34]. Additionally, as of September 2023, the proposed I-24 MOTION system produces vehicle trajectory data during morning rush hour (6:00AM to 10:00AM) each day. The resulting trajectory data is made publicly available for research purposes. Any given day of trajectory data is larger than all existing trajectory datasets (in terms of observation area length,

duration of data recording and number of vehicles) and over time the I-24 MOTION trajectory data will be the only publicly available, continuous vehicle trajectory dataset except possibly among private entities.

1.2 Dissertation Organization

The remainder of this disertation is organized as follows: Chapter 2 provides a review of related works, situating this work among existing literature. Chapter 3 describes the I-24 MOTION instrument, starting with the initial feasibility tests performed to assess the concept of a large-scale video-based trajectory generation testbed, and ending with a description of the I-24 MOTION system as constructed and an overview of the first trajectory dataset released from the instrument. Chapter 4 details the fast single-camera object tracking methods developed to enable efficient processing of video from the I-24 MOTION camera system. Chapter 5 describes the extensions proposed to make this method suitable for tracking objects across multiple cameras in 3D space. Chapter 6 describes a ground-truth 3D multi-camera dataset developed using video data from I-24 MOTION for purposes of training and benchmarking object detection and tracking algorithms. Chapter 7 describes a much larger video dataset with sparse vehicle annotations developed to allow for extremely long term object tracking performance benchmarking. Finally, Chapter 7.5 concludes the dissertation with perspectives on the future of the I-24 MOTION instrument and on trajectory data efforts more generally. Lastly, don't miss the Appendices! They provide numerous mathematical formulations, derivations, additional results, painstakingly thorough analyses of error modalities for various forms of data, discussions of privacy considerations, and experimental details.

2. Related Work

This literature review explores existing works in similar domains to this dissertation. Namely, it provides an overview of existing traffic observation testbeds and datasets including vehicle trajectory data in Sections 2.1 and 2.1.1. The core contributions of the dissertation work rely on joint object detection and object tracking methods. Approaches to each constituent problem are briefly discussed in Sections 2.2 and 2.3, and a comprehensive overview of online joint object detection and tracking methods is conducted in Section 2.4. Tracking based solutions for the traffic task of vehicle turning movement counting are explored in Section 2.5. Approaches to multiple camera tracking are reviewed in Section 2.6. Lastly, existing single and multiple camera multiple object tracking datasets are reviewed and summarized in Section 2.7 and Section 2.8.

2.1 Existing vehicle testbeds

A *vehicle testbed* consists of a section of roadway equipped with sensors that collect data on the vehicular traffic through that portion of roadway. These sensors generally provide a richer set of data than is obtainable on an un-instrumented section of roadway. Common sensors utilized include cameras, *Light Detection and Ranging* LIDAR sensors, radar (speed estimation) units, and *Dedicated Short Range Communication* (DSRC) devices allowing vehicle-to-vehicle and vehicle-to-infrastructure communication. The collection of this data allows for finer-grained analysis of vehicles and traffic than could otherwise be obtained. Existing *closed course* and *open road* testbeds already address some critical emerging research needs [96]. Closed course testbeds, such as the American Center for Mobility [97], MCity [98], GoMentum Station [99], and Suntrax [100], have the distinct advantage of being capable of hosting experiments and data collection for cutting edge technologies and techniques including those under active research and development. By testing in highly controlled settings, they can assure safety and eliminate external factors such as unpredictable drivers and road conditions that can confound experiments. Because of the motivating objectives of closed course testbeds, they can be limited in their ability to test in real traffic conditions with regular drivers encountered on public roads. Open road testbeds exist in many forms on a variety of road types; examples include the Minnesota Traffic Observatory [101], The Ray [102], the California Connected Vehicle Test Bed [103], Ann Arbor Connected Vehicle Test Environment [104], and Providentia [105]. Table 2.1 summarizes existing vehicle testbeds and the uses their data enables.

Testbed	**Location**	**Sensors**	**Open Road?**	**Use**
ACTION [106]	Tuscaloosa, AL	DSRC, Cameras	Open road	CV, V2I
M-City [107]	Ann Arbor, MI	DSRC, Cameras	Closed course	AV
The Ray [108]	Interstate 85, GA	DSRC	Open road	CV, V2I
California CV Testbed [109]	Palo Alto, CA	DSRC	Open road	CV, V2I
Gomentum [99]	Concord, CA	LIDAR, DSRC, Cameras	Closed course	AV, CV
ACM Proving Grounds [110]	Ypsilanti, MI	DSRC	Closed course	AV
SunTrax [111]	Orlando, FL	DSRC	Open road	V2I
AACTVE [112]	Ann Arbor, MI	DSRC	Open road	V2I
Providentia [105]	Munich, DE	Radar,Cameras	Open road	Trajectories
Minnesota Traffic Observatory [101]	Minneapolis, MN	Radar	Open road	Trajectories
Lower Saxony Testbed [79]	Braunschweig, DE	LIDAR, DSRC, Cameras	Open road	Trajectories, CV
Zen Traffic Roadways [80]	Osaka, JP	Cameras	Open road	Trajectories, CV, AV

Table 2.1: Existing vehicle testbeds. *CV* indicates connected vehicle technology testing, *V2I* indicates vehicle to infrastructure communication testing, *AV* indicates autonomous vehicle testing, and *Trajectories* indicates the testbed produces speeds and positional data for vehicles on the roadway.

2.1.1 Vehicle Trajectory Data

Vehicle trajectory data and consists of vehicle positional data for each vehicle within a traffic stream. Generally, positional accuracy on the order of 1-foot is required to allow accurate calculation of derivative quantities such as velocity, acceleration and steering angle. Figure 2.1 shows an example of vehicle trajectory data. This

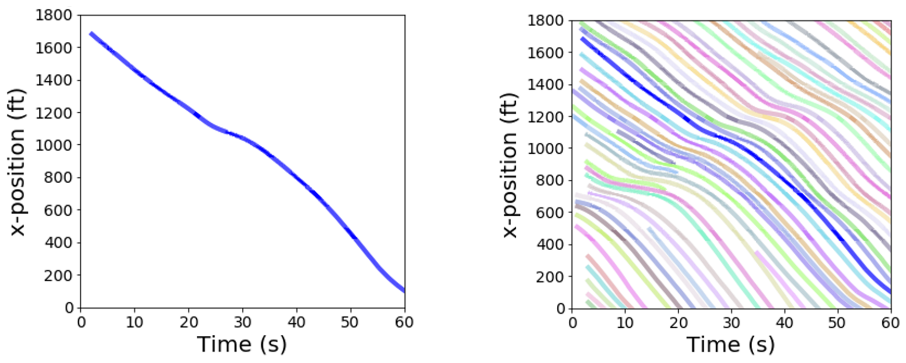


Figure 2.1: Vehicle Trajectory Data. (left) A single vehicle trajectory. (right) Vehicle trajectories for every vehicle in one lane of traffic.

data contrasts with traditional traffic sensing approaches in that it is at once macroscopic (i.e. contains information about the global traffic flow such as throughput, vehicle density and latency) and also microscopic (i.e. contains the precise positional and derivative data for each vehicle). Such data is required for myriad traffic analysis and modeling applications, yet sources are limited. Of existing vehicle testbeds (see Table 2.1, only the Minnesota Traffic Observatory [101], Providentia [105], the Lower Saxony Testbed [79], and Zen Traffic Roadways [80] are capable of producing vehicle trajectory data. The first two report average positional errors (RMSE) of over 3 meters, so only the latter two Lower Saxony Testbed provides fine-grained vehicle trajectories. However, data from [79] is not currently made publicly available, so the testbed use in research is limited and the quality of the data cannot be assessed. Given this data shortage, much research requiring vehicle trajectory data relies on the NGSIM dataset [49], collected almost 2 decades ago and known to contain large vehicle positional errors [50].

2.1.2 Emerging Observation Technologies

In a parallel thread, significant research has been devoted to the computer vision tasks of *object detection* (locating relevant objects within an image) and *object tracking* (associating distinct objects in video frames across time). Especially in the past 10 years, rapid progress has been made in the use of modern hardware [59], neural network architectures [60–63], and massive-scale image datasets [64, 65] to fit accurate object detection algorithms. Approaches for extracting vehicle trajectory data utilizing these techniques have been proposed. For example, the work [113] proposes a method to detect vehicle 3D rectangular prism bounding boxes using background subtraction and blob segmentation, relying on automatic parameter extraction of the scene homography proposed in [114]. The work [115] uses this data to train a *convolutional neural network* (CNN) to produce the same data without the need for scene-wide calibration. In [116], 2D object detectors are used to estimate vehicle positions on the road plane (the ambiguity of vehicle position within a 2D bounding box is not fully addressed). [117] uses ground plane projection of vehicle pixels from multiple cameras to estimate the vehicle's position, validating with turning movement counts. Other solutions rely on re-identification of 2D tracked objects, without addressing 2D annotation position ambiguity [118, 119]. Other methods utilize instance segmentation networks [77, 120] on traffic scenes with little occlusion. A few approaches [121, 122] avoid object detection by measuring object presence in *longitudinal scanlines* along each roadway lane, but occlusion and lane changes pose difficult challenges in this problem formulation. In theory, such methods promise to address the shortage of trajectory data.

These advances, along with the increasing prevalence of aerial drones, have enabled recent research efforts to revisit the task of vehicle trajectory extraction and make marked advancements to the state of the art. The HighD, [66], ExiD [67], AUTOMATUM [68], and HIGH-SIM [69] datasets all utilize aerial imagery shot from either drone or helicopter-mounted cameras to produce complete highway vehicle trajectory data, and the *Third Generation Simulation* (TGSIM) [70] is a similar in-progress effort designed to capture trajectory data containing deployed automated vehicle technologies. Similarly, the pNEUMA [71], inD [72], rounD [73], OpenDD [74], Interaction [75] and CitySim [76] datasets utilize drones or swarms of drones to study complex urban vehicle and pedestrian interactions in more detail. High aerial fields of view make modern image segmentation algorithms [77] well posed for vehicle tracking in these contexts, but these methods are temporally limited by the relatively short battery life of drones (generally under an hour) and the requirement for human pilots. Some additional works utilize sensor-equipped vehicles [123, 124] or GPS data [125, 126] to collect vehicle trajectories, but these technologies do not provide data for non-instrumented vehicles. A persistent shortage of trajectory data in many cases requires that researchers rely on models such as TransModeler [127, 128] or SUMO [129] to simulate the interactions between drivers at scale, which only approximate human driver behavior.

2.2 Object detection

Nearly all top-performing object detection methods rely on *convolutional neural networks* (CNNs) for feature extraction from an image [59]. Each of these networks consists of a *backbone* that extracts a meaningful, lower-dimensional set of features useful for the derivative task of object detection from the extremely high-dimensional set of inputs for the object detection task (pixel intensity values), and one or more *heads* that use this feature set to produce the desired outputs (namely object positions as 2D bounding boxes and per-class confidences). The backbone is composed of convolutional neural network layers, pooling layers which aggregate information to reduce the representation dimension, and custom architecture-specific layers designed to increase task performance. Almost universally, CNN architectures designed for object classification such as [60] are used as the backbone because these models can be pre-trained on huge datasets for object classification [64] and empirically, features useful for classification are also useful for detection.

A few common structures exist for the detection and classification heads. Anchor-based methods explicitly assign an output to each region and potential object size in an image, and backbone features for that image are mapped to these outputs by either fully connected or convolutional neural network layers. Anchor-free methods instead map backbone features to a heatmap of object locations with convolutional neural network layers and regress object sizes from this heatmap. These methods generally require custom pooling layers within the backbone architecture that better convey keypoint information about objects through convolutional layers. Within both anchor-based and anchor-free methods, a distinction exists between one-stage and two-stage detection methods. In one-stage methods, the final outputs are directly regressed as offsets from the anchors. One-stage anchor-based methods include YOLO [61] and its various derivatives [130], [131], Retinanet [132], and SSD [133], and one stage anchor-free methods include CornerNet [134] and CenterNet [63]. In two-stage methods, *region proposals* or *keypoint proposals* are first regressed. These proposals are used to crop local features from the associated portions of the image, and each region is subsequently processed by a second regression head to output the final object detections and classifications. Two-stage anchor-based methods include Faster-RCNN [135] and Evolving Boxes [136]. Recent object detection works have utilized additional architectures for detection and classification heads. Segmentation models such as Mask-RCNN [77, 137] have also been adapted to perform bounding box-based detection as they provide a richer output from which the object detection outputs can be trivially obtained. Attention networks [138] or transformer networks [139, 140] have also been proposed for object detection. Additionally, object detection approaches have also been bolstered by adding additional awareness of foreground and background [141], by use of a Detection models are generally designed such that they can process (640×480 pixel) frames from benchmarking datasets such as COCO [65] quickly, but speed often degrades to or below 30 frames per second on a GPU for the best-performing object detection methods for frames of modest size (e.g., 960×540 [1], 1392×512 [81], and 1920×1080 [82]).

2.2.1 Monocular 3D object detection

Monocular 3D detection methods seek to generate a set of 3D bounding boxes in 3D space based on a single camera image. One early work is Mono3D [142], which generates rich 3D proposals with the assumption that vehicles locate on the ground plane and then scores the boxes with contextual information and size, location, and shape priors. Likewise, [143] generates 3D proposals and ensures that feature map computations are orthographic such that objects further away and occupying fewer pixels do not occupy less of the final feature map space. In [144], detection and object tracking are accomplished by directly regressing 3D coordinates, but anchor boxes are generated in 2D image-space (thus the scene homography is implicitly learned during training).

2.2.2 Viewpoint-Agnostic Monocular 3D Detection

Viewpoint agnostic monocular methods can roughly be divided into two categories: i.) methods that regress 2D bounding boxes or segmentations along with augmenting outputs and utilize homography constraints to subsequently predict 3D outputs, and ii.) methods that regress 3D projections of keypoints or bounding box corner points.

In the first category, Deep3DBox [145] predicts a 2D bounding box, the observation angle, 3D object size, and object 3D center position (in the image) from the features enclosed by the 2D bounding boxes, as the bounding box can subsequently be fit by the constraint that its 2D projection falls within the 2D bounding box. Shift R-CNN [146] and Cascade Geometric Constraint [147] leverage the fact that 4 vertices of the 3D bounding box must lie on the 2D bounding box. The main drawback of these models is that they rely on accurate predictions of 2D bounding boxes. Errors in 2D bounding boxes compound in the 3D prediction. Likewise, [148] utilizes a 2D bounding box and manipulates a simplified 3D vehicle model to optimize the 3D object position within the bounding box. 3D-RCNN [149] takes additional segmentation inputs and generates a compact 3D representation of the scenes. It exploits class-specific shape priors by learning a low-dimensional shape-space from collections of CAD models.

Most recent methods fall in the latter category, representing vehicles as polyhedrons or 3D bounding boxes. Mono3D++ [150] represents a vehicle as 14 keypoints and learns the 2D keypoints using EM-Gaussian method. MonoRCNN [151] is built upon Faster R-CNN and adds 3D attribute and distance heads to recover 3D bounding boxes. The heatmap concepts proposed by CenterNet [63] inspired many monocular 3D detection models because this model's structure is well-suited to keypoint regression. RTM3D [152] uses CenterNet-based structures to regress nine keypoints of the 3D bounding box corresponding to the eight corners and the center of the 3D cuboids. RTM3D also regresses distance, 3D box dimension, and orientation of vehicles, then solves an optimization for the best-fitting bounding box in 3D space for each object. Likewise, Monocon [153] and Monoflex [154] are built upon CenterNet. Monoflex directly predicts 3D dimensions and orientations and incorporates multiple depth estimation methods to increase accuracy. In [155], no scene information is ever used, and instead the vanishing points and, thus, scene homography are directly computed from output 3D bounding boxes (albeit in a traffic monitoring context).

2.2.3 IoU Loss in Object Detection

L1 and L2 losses are widely used in object detection models but ignore the shape of bounding boxes and are easily influenced by the scales of boxes. Conversely, IoU encodes the shape properties of objects and is invariant to the scale. Thus, IoU-based loss formulations have achieved good performance in object detection. In [3], Generalized Intersection over Union (GIoU) loss is proposed to provide better convergence for non-overlapping bounding boxes. The authors incorporate GIoU loss into YOLO v3 [130], Faster R-CNN [156], and Mask R-CNN [77] and show a consistent improvement in their performance on popular object detection benchmarks such as PASCAL VOC and MS COCO. [157] similarly incorporates the distance between bounding boxes to aid convergence in non-overlapping cases. In [158], the authors introduce Complete-IoU (CIoU) loss to consider three geometric factors: overlap area, normalized central point distance, and aspect ratio. CIoU loss is used in YOLOv4 [131] and leads to notable gains of average precision (AP) and average recall (AR). In [4] and [159], IoU loss is defined for two rotated bounding boxes into several 3D object detection frameworks. which leads to consistent improvements for both bird-eye-view 2D detection and point cloud 3D detection on the public KITTI benchmark [160].

While promising, these IoU loss variants are not suitable for the keypoints of 3D bounding boxes projected to the image plane. No work yet analyzes the performance of incorporating IoU loss into viewpoint-agnostic monocular 3D detection frameworks.

2.3 Multiple object tracking

The multiple object tracking task seeks to produce the position and unique identity for each distinct object appearing across a sequence of images or video. The vast majority of approaches consider this task given an input set of object detections for each video frame from any arbitrary object detection algorithm, known as the *tracking by detection paradigm*. Figure 2.2 provides a graphical summary of this task.

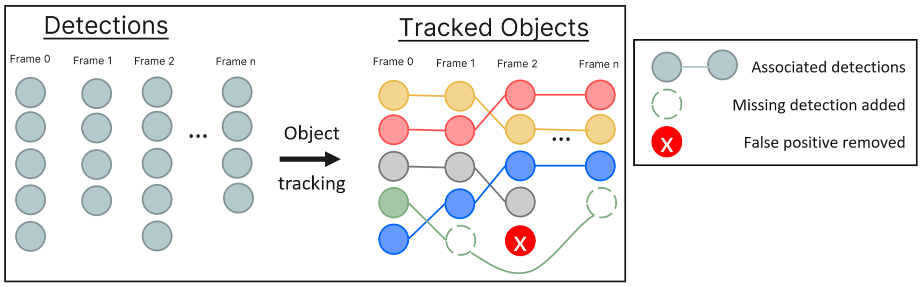


Figure 2.2: Multiple Object Tracking Problem. Detections (circles) are assigned unique IDs (colors). Missing detections (dashed circles) must be imputed, and detection false positives (red X) must be removed.

Multiple object tracking has been comprehensively reviewed [161–163]. In this review, we provide a breakdown of popular methods along 3 informative lines: the online or offline nature of the method, the fundamental problem formulation solved, and the auxiliary models of object information that are incorporated. Each is discussed in the following subsections.

2.3.1 MOT processing mode

Online multiple object tracking methods including [5, 60, 86, 89, 164–173] do not make use of future information when predicting the set of object states at any time. That is, the predicted set of objects is output for frame n after seeing detections from frames $0,...,n$ but before seeing detections for frame $n+1,...,N$. By contrast, offline methods [174–185] make use of information from the entire set of object detections for all N frames to produce tracked objects for each frame in the sequence. A third category, called *near-online* methods [186–189], consider only information from frames $0,...,n+k$ to produce object tracking outputs for frame n, where the lag k is relatively small relative to the overall number of frames N.

2.3.2 MOT problem formulation

As noted in [161], it is difficult to comprehensively and precisely categorize the corpus of MOT problem formulations into a distinct and meaningful taxonomy. However, the vast majority of algorithms can roughly be described as either *graph-based methods*, *energy-based methods*, or *probabilistic representation methods*.

2.3.2.1 Graph-based MOT methods

The MOT problem is well-formulated as a graph-based data association problem [190]. In this formulation, each detection is represented by a node, and edges are assigned to associate detections into object tracklets. The MOT problem has been formulated as a maximum weight minimum-cost flow problem [174, 176, 187–189] subject to exclusion and unit flow constraints (a maximum of one edge into and out of a node, each with unit flow), a maximum weight independent set problem [175, 177, 186] (maximizing edge weights between detection nodes belonging to the same object / set), a conditional random field graph problem [178, 185] associating small object tracklets with one another, and, most often, a bipartite matching problem [5, 60, 89, 166–169], where the optimal matching of existing tracked objects to new detections is selected on the basis of edge weights. These graph problems can be solved optimally with existing algorithms [5, 166, 176], with proposed efficient optimization algorithms that efficiently explore the space of possible solutions [178, 188] or with greedy approaches [87, 187]. Importantly, each graph-based method relies on defining meaningful edge weights (alternately referred to as the cost or affinity) between pairs of objects or detections. Such weights are generally defined as a combination of various information sources about the detections, including geometry/position, appearance, physical dynamics, etc. Methods may ensemble combine these various sources of information analytically [87, 89, 170, 181, 191–194] or else utilize additional convolutional layers [195, 196], recurrent neural networks such as LSTMs [186, 197–201] or graph neural networks (GNNs) [168, 169, 189, 197, 202, 203] to predict affinity by modeling the interactions between these various sources of information across co-occurring objects and across time. These various sources of information are discussed in more detail in Section 2.3.3.

2.3.2.2 Energy-based MOT methods

Energy-based methods consider an exhaustive state space of continuous object positions and discrete object identity assignments [180–185]. Each state within this space is scored according to its energy, which is generally defined according to some combination of information sources discussed in section 2.3.3. Then, the optimal state is determined by optimization / minimization of this energy term. In practice, the search space is highly non-convex, so efficient algorithms are proposed to explore this space and converge to an optimal solution more quickly.

2.3.2.3 Probabilistic representation MOT methods

Probabilistic methods maintain a probabilistic distribution of possible object positions for a given time, rather than a deterministic representation. This distribution may consist either of a "search tree" of discrete possible object assignment sets, as in multiple-hypothesis trackers [164, 165, 186], or a continuous distribution of object locations and likelihoods, as in filtering-based methods [172, 173]. Note that while energy-based methods explore a space of possible hypotheses, they solve an optimization to finally represent object positions deterministically and generally in an offline or semi-online manner, whereas probabilistic methods are online or near-online and maintain a non-deterministic representation of each object during tracking. In each case, a necessary step to produce object tracking outputs is the determination of the maximum likelihood estimate of object positions from the distribution of possible states.

2.3.3 Incorporation of information into MOT problem

A number of sources of information can be incorporated into multiple-object tracking. Major categories of information are briefly discussed here:

1. **Position/Geometry** - Object representational geometry such as object center [5] or bounding box overlap [166] are used to compare existing objects and detections.

2. **Appearance Information** - Object appearance is encoded as a vector or set of features generated from constituent image pixels. These features are sometimes generated with manually designed feature extractors such as color or object contour shape extractors [181, 185, 204]. Most recent approaches use CNN-based appearance embeddings learned to minimize embedding distance between embeddings of the same object and maximize embedding distance between embeddings of different objects [60, 89, 167, 171, 193, 205, 206]. This distance is generally calculated as cosine distance between the embedding vectors.

3. **Physical Dynamics** - Object motion is modeled, often with a constant velocity assumption [86, 88, 194, 207], or with higher order dynamics considered [185, 208]. Filtering methods are also used to optimally combine motion models with object measurements (detections) [5, 60, 202]. Optical flow prediction, [90], *Long-Short term memory networks* (LSTMs) [186, 198, 199, 202], and velocity regression during detection [86, 87] have also been used to model object motion implicitly with neural networks.

4. **Interaction Models** - for instance, individuals moving in a group should maintain the same approximate configuration. [161] provides a more comprehensive overview of interaction models.

5. **Exclusion Models** - the same objects should not be associated with the same or very similar detections [181], and should not occupy the same physical space at the same time [209].

6. **Occlusion Models** - a variety of methods are considered for gracefully handling instances where objects are temporarily obscured. Common approaches include propagating object locations into the future using motion models [5, 86], or using appearance embeddings to re-match objects to with a set of disappeared objects when they become visible again [194, 210]. Methods may additionally model when objects are occluded and exclude these objects from appearance and position updating [209, 211–213], or stitch tracklets for the same object together with a single object tracker post-processing step [214].

2.3.4 Multiple object tracking evaluation

Much work has been done to create evaluative metrics for multiple object tracking that richly capture the performance of tracking algorithms in terms of a variety of desired qualities [215–218]. From among these metrics, *Multiple Object Tracking Accuracy* (MOTA) is overwhelmingly the most often used aggregate metric to assess tracking accuracy. Adopting the notation from the proposing work [216], MOTA weighs 3 intuitive types of tracking errors: *false positives* (fp), in which a predicted object position in a frame cannot be matched to a suitably similar ground truth object position in that frame, *false negatives* or *misses* (m), in which a ground truth object position for a frame cannot be matched to a suitably similar predicted object position in that frame, and *mismatches* (mme), in which the predicted identity matched to a ground truth object trajectory changes (either because the object was lost and re-tracked as a new object, or because two tracked objects switched identities). The aggregate metric compares the total of these errors relative to the total number of ground truth object positions g for a frame, and is calculated over all frames (indexed by n) as:

$$MOTA = 1 - \frac{\sum_n m_n + fp_n + mme_n}{g_n}. \tag{2.1}$$

Recently, the *Higher Order Tracking Accuracy* (HOTA) metric has been proposed and gained popularity because it expresses localization accuracy, detection precision and recall, and association accuracy in a single score [218].

2.4 Joint Detection and Tracking

The previous section discussed methods for object detection and object tracking, respectively. Generally, object detection is tracking-agnositc; that is, object detection is performed without regard for how the outputs will be used during tracking and without using any information which might be gained from the tracking context. However, recent works aimed at tracking objects through video have proposed *joint* approaches that do utilize the tracking context in various ways. This approach has a strong intuitive basis; generic object detectors assume that the output distributions for consecutively processed images are independent of one another, but in reality the positions of objects within closely consecutive video frames are highly correlated. Joint detection and tracking methods seek to leverage this additional information to increase tracking performance (generally in terms of accuracy). Figure 2.3 provides an overview of the joint tracking and detection framework.

The following subsection provides an overview of notable approaches to the joint detection and tracking problem, broken down into four primary categories of tracking information utilization. Note that despite this categorization, some methods use multiple categorical techniques for combining the detection and tracking tasks. Unless otherwise noted, each joint method is online. This is because the information from previous frames output tracklets is explicitly required for future frames. Within the online framework, the vast majority of approaches use bipartite matching formulation, but some solve other graph-based problem formulations.

2.4.1 I.) Additional object detection outputs

Methods in this category are weakly joint; the sequential nature of video frames is not utilized to improve object detection performance; however, these methods extend object detection models to produce additional outputs that are useful for the object detection task; thus they incorporate the object tracking task into object detection. Primarily, methods output object appearance embeddings for subsequent appearance-based affinity computation as well as object bounding boxes in a multi-task learning framework [167, 170, 192, 193, 202, 205, 219]. By producing these outputs in the same model (as opposed to computing appearance features after object positions and classes as in previous works), information from each task can be utilized to improve performance on the other.

A number of other tracking-specific output heads are added to object detectors in other works. (Note that these outputs are accompanied by additional usages of the tracking context, so these works are described in more detail in following subsections. Some methods output rasterized heatmaps of object location likelihoods [87, 167, 220, 221], and others output object displacements (predicted motion) in lieu of an explicit motion

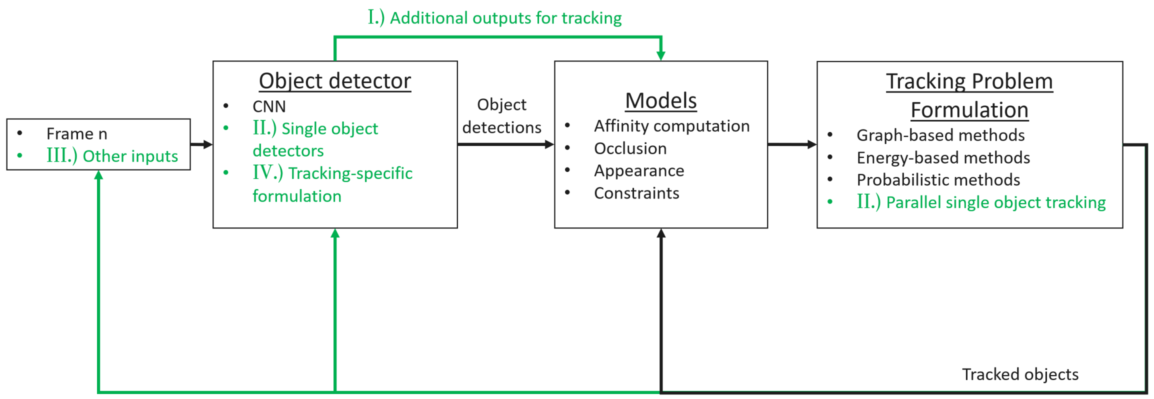


Figure 2.3: The joint detection and tracking paradigm. Items in green are unique to the joint paradigm and are not present in the tracking-by-detection paradigm. The joint paradigm utilizes tracking information in one or more of four distinct ways: I.) Object detectors output additional information useful for tracking problem formulations (i.e. data association). II.) Single object detection and tracking methods are utilized in parallel to perform multiple object tracking. III.) Additional tracking-specific information is input to the object detector. IV.) The object detector is re-architected specifically for joint detection and object tracking.

model [87, 220, 222, 223]. Finally, transformer-based joint methods output ”queries” or latent representations of object appearances for recursive usage by the transformer model on the following frame [212, 224, 225].

Lastly, a few methods output practically no new information at all, but opt to output all object detector outputs rather than thresholding these values based on predicted object confidence. Generally, only detections with confidence higher than some threshold are saved as outputs, and furthermore overlapping detections are pruned. In [226] and [227], all detections are saved, and existing tracked object information is used to select from this complete set of detection outputs rather than only the high-confidence subset.

2.4.2 II.) Formulation as parallel single object tracking problem

A large number of existing works such as [228–232] explore the task of detecting and tracking a single object in video, usually with the object of interest manually identified on the first frame. Recent joint detection and multiple object tracking works have made use of highly accurate *single object trackers* (SOTs) for multiple object tracking, and can be roughly divided into three sets of methods.

The first set of SOT methods utilizes multiple single object trackers in parallel [210, 233, 234]. These methods make use of object detectors to initialize new objects at each frame or at the beginning of the video sequence [234], and subsequently uses a single object tracker to track each unique object.

A second set of SOT methods ensembles the results from single object trackers and object detectors at each frame, selecting the best outputs to match to each tracked object [209, 211, 223, 235]. In a similar vein, [214] runs a tracking by detection framework but subsequently revisits each frame with single object trackers to stitch together fragmented trajectories [214].

Finally, a third set of SOT methods compute shared features on a frame, and then utilize custom neural network heads that explicitly associate their outputs with a single object each. In [212], a separate attention head is added for each object, which outputs a detection and a query that is input to that object head at the next frame (sharing appearance and positional information across frames). [86, 90] map each object prior to a single region proposal or anchor box, and the resulting regressed bounding boxes are then implicitly associated with the corresponding objects. Transformer-based methods [224, 225] pass a query per object to the object detector, and the corresponding outputs for each query are likewise implicitly associated with the corresponding objects.

2.4.3 III.) Additional inputs to object detector

Methods in this category are trained with additional or alternate inputs that are not available in a pure object detection context, but are available in a tracking context. Trained object detectors make use of the additional information offered by these inputs (especially with respect to localizing objects in motion, maintaining object persistence across frames, and gracefully handling temporary object occlusions) to boost output detection performance. Additional inputs to object detection networks in joint methods generally fall into two classes.

In the first class, two [87, 89, 194, 207, 220–222] or more [192, 203, 208, 234] consecutive frames are passed as inputs to the detector.

In the second class, predicted existing object locations or *priors* are passed as inputs to the object detector. In [87] and inspired works [89, 220, 221], priors are input as a heatmap image of object center locations. In transformer-based approaches such as [224, 225, 236], objects embedding vectors or search *queries* are passed into the network.

2.4.4 IV.) Object detector architecture reformulation

The previous class of methods utilized tracking information in the detection context with additional inputs to off-the-shelf object detectors. In contrast, this class of methods uses a different formulation of the object detector architecture to explicitly utilize tracking information. There are three major sets of approaches in this class.

First, some methods use object priors to select the best matching anchor box from among one-stage detector anchor boxes [90] or to explicitly add region proposals linked to each active object in a two-stage detector network [86, 223]. Similarly, [171] weights each region proposal both based on predicted confidence (as in normal two-stage detectors) but also based on overlap with object priors.

Second, other methods consider sets of anchor boxes or region proposals spanning multiple frames (i.e. a region proposal includes a bounding box for an object in each frame). These methods necessarily also take multiple frames as input. Some methods consider anchors spanning pairs of frames [207, 222] while other consider longer object ”tubes” spanning arbitrarily many frames [88, 203].

Lastly, some methods utilize 3D convolutional neural networks, treating a set of consecutive frames as a 3D volume and learning neural network features spanning multiple frames. [88] and [213] combine this 3D convolutional structure with 3D (across multiple frames) region ”tube” proposals (and subsequently utilize 3D tube intersection for bipartite matching affinity) , while [203] and [234] opt for 3D convolutions but single frame region proposals.

Table 2.2 provides a summary of the four discussed classes of joint tracking methods, as well as the major sets of approaches within each class.

2.5 Turning movement counting

One common derivative usage of multiple object tracking methods in the domain of traffic monitoring is for vehicle turning movement counting at roadway intersections. Each turning movement is describable by an origin roadway segment and a destination roadway segment, and vehicle tracking associates an origin and destination with that vehicle. General approaches to this problem can be categorized into single movement and multiple movement vehicle counting.

2.5.1 Single movement vehicle counting

The task of single movement vehicle counting or counting of vehicles passing a fixed line generally requires object detection, as well as object tracking to avoid double-counting the same vehicle in multiple frames. As noted in [237], single movement vehicle counting is still a challenging task in cases where camera field of view creates extremely high overlap between vehicles. In [238], an early algorithm for object counting is proposed utilizing background subtraction, blob fitting to cluster pixels into objects, and Kalman filtering to track vehicles across frames. [239] utilises Gaussian Mixture models for clustering background-subtracted pixels and compares each resulting cluster’s convex hull area to its contained bright pixel area to explicitly predict object occlusion. [240, 241] use CNN-based object detectors and the Kanade-Lucas-Tomasi feature tracker to track and count objects. Similarly, [242, 243] utilize CNN object detectors and Kalman filtering for object tracking through the movement of interest, and [244] combines a cascade feature-based CNN

Joint Context	Works
I.) Additional Detector Outputs	
Re-identification Embeddings	[170], [167], [205], [89], [193], [194],[219], [192],[202]
Object Displacement Map	[223],[222]
Object Predicted Motion Offsets	[87], [220]
Heatmap of object locations	[87],[89],[221],[220]
Object Latent States	[212]
Future object positions	[86]
All (un-thresholded) detector outputs	[227], [226]
II.) Parallel Single Object Tracking	
Separate Tracker per Object	[210],[209],[233]
Shared CNN Features	[212],[224] ,[86], [90],
Ensemble of Detector and Single Object Trackers	[214], [211], [235], [234], [223]
III.) Additional Inputs to Detector	
Multiple Frames	[170], [89], [222], [207], [87], [192], [208], [203], [234], [88] [194],[221],[220]
Object Prior Heatmaps	[89], [87],[221],[220]
Object Embeddings as Queries	[236], [224], [225]
IV.) Modified Detector Formulation	
Anchors/ Proposals Across Multiple Frames	[207], [222], [208], [88]
Anchor / Proposal Selection using Object Prior	[86, 90, 171, 223]
3D Convolutional Neural Network	[203, 234]

Table 2.2: Joint Tracking Methods by Usage of Tracking Context.

with IOU tracking [166]. [245] also utilizes a weakly defined homography transformation into real-world coordinates to estimate each tracked vehicles length and inform vehicle classification. [246] makes use of foreground and background information to drastically reduce the feature space relative to image pixel-space before regressing object locations. [247] does not track objects, but instead maintains occupancy counts for several regions with the frame to logically determine when a vehicle should be counted.

2.5.2 Multiple turning movement counting

The task of multiple turning movement counting is distinct from single-movement vehicle counting in that a movement uniquely defined by an object's origin and destination must be predicted for each counted object. As with single movement counting, multi-movement vehicle counting is still a challenging task especially when realtime performance is required. [248] utilizes a neural network to predict multiple vehicle turning movement counts at intersections given only aggregate approach traffic volume, which could effectively turn single-movement algorithms into multiple-movement counting algorithms, but this approach is not widely used. Nearly all algorithms for vehicle turning-movement follow the detect-track-count paradigm, where objects are detected in each frame, tracked across frames, and tracklets are subsequently categorized into turning movements, though [249] instead utilizes a joint tracking and detection method, Tracktor [86], and [250] instead directly regresses vehicle counts from an input video using a *Long-Short Term Memory* (LSTM) neural network to incorporate temporal information into the task. Most approaches for counting vehicle movements from trajectories utilize trajectory passage through unique sets of regions with the camera field of view to uniquely identify the relevant turning movement [249, 251–255], directly compare trajectories to canonical turning movements from each possible movement category [245, 256–258], or do some combination of the two [259, 260]. While many approaches in the first category require only a source and sink region to uniquely define a movement, some methods utilize larger sequences of regions to help distinguish between turning movements that occupy similar areas within a camera field of view [252, 254]. In the second category, the *longest common subset* (LCSS) shared by object trajectories and canonical turning movements is often used to assign turning movements to trajectories [256], but K-nearest neighbors clustering [257], scale-normalized trajectory similarity [258] and Hausdorff distance [245] are also used. [258] also scores each turning movement in terms of stability, completeness, and proximity to each object's trajectory, and smooths out anomalous points in each trajectory. [260] performs segmentation on trajectories and compares segments to known turning movement segments.

2.6 Multiple camera, multiple object tracking

Multiple camera multiple object tracking (MCT) seeks to associate objects tracked in a set of individual camera views with one another, or else to output tracked objects detected in the set of cameras natively in a shared space (e.g real-world ground plane). The vast majority of these methods can be separated into 2 classes: methods that perform single camera tracking followed by multi-camera clustering of trajectories, and methods that perform single camera detection, cluster detections across cameras, and track in a unified space.

2.6.1 Input fusion

These methods performs object detection utilizing frames from all cameras simultaneously [261–264], subsequently perform object tracking on the single set of detections output.

2.6.2 Single camera tracking, multi-camera trajectory clustering

Generally, these methods formulate cross-camera trajectory clustering as a graph-based problem such as maximum weight clique [265, 266], or else solve the problem by greedy approximation [118, 267–273]. The affinity between two trajectories is generally computed as a combination of vehicle appearance embeddings and spatio-temporal similarity (based on 3D space distance in two cameras at the same time) [266, 267, 269, 271–273]. [118] also uses license plate recognition features when possible to aid in trajectory linking. Some methods identify shared spatial zones across multiple cameras and require that clustered trajectories visit the same sequence of zones [271], or else utilize a camera link model which specifies a limited set of possible object paths through a set of camera views [267, 269, 270, 272, 273]. [268] also notes that matched trajectories must both cross a line shared between both cameras at the same time, using this information to reduce the matching search space. [274] formulates the trajectory linking problem as a multiple hypothesis tracking problem, using trajectory appearance and motion to estimate a hypothesis' likelihood. Lastly, [275] tracks objects in single camera views at a time with Kalman Filter representations and "hands off" objects between cameras when they near the extents of camera fields of view.

2.6.3 Single camera detection, detection clustering, and shared-space tracking

Several methods in this category detect objects by background subtraction, and project the detected pixel blobs into a unified space where they can be concatenated by mixture of Gaussian methods [276, 277]. The resulting shared-space detections are then tracked with simple methods such as in [5] or else by probabilistic hypothesis density filtering [277]. [278] uses a similar approach specifically for pedestrian tracking but searches for human head shapes on a plane offset from the ground plane. Other approaches rely on off-the-shelf object detectors, combining detections using hierarchical clustering methods that utilize appearance and location information [279, 280]. [281] uses a Bayesian belief network to assign new detections to shared space tracked objects modeled with Kalman filters.

2.7 Multiple object tracking datasets

This section briefly reviews existing datasets for multiple object tracking. The representation of objects (2D or 3D) and the context of each dataset is noted.

2.7.1 Single Camera MOT Datasets

The task of single camera 2D *multiple object tracking* (MOT) is well-studied in varied contexts, including pedestrian and vehicle tracking from stationary and moving cameras (MOT16, 17 and 20) [82, 217], tracking from drone footage (VISDRONE) [282], and traffic monitoring (UA-DETRAC) [215], as well as arbitarary object class tracking [283]. 3D single camera (or stereo camera for depth) multiple object tracking is also well-addressed within the domain of *autonomous vehicle* (AV) or ego-vehicle data (KITTI, Waymo Open-Drive, and NuScenes) [124, 284, 285]. Data annotation in this context is aided by rich LIDAR data from on-vehicle sensors. As a result, effective methods for 3D tracking in this context have been proposed, representing vehicles as 3D rectangular prisms in 3D space [286] or image space [152], 3D voxel patterns [287], or CAD-derived shape models [148].

Rich 3D data in the traffic monitoring (overhead traffic camera) domain is sparse, in part because LIDAR sensors are not collocated with cameras to aid in annotation. Only the BoxCars116k dataset [115] provides (automatically generated) 3D monocular bounding boxes. Thus, research on 3D vehicle tracking from overhead cameras must use simulated or partially synthetic data [288–290].

2.8 Multi-camera MOT datasets

Few *multiple camera multiple object tracking* datasets exist, mostly in the context of pedestrian tracking. The Duke-MTMC dataset [291] and CamNeT [265] each associate 2D object tracklets for pedestrians across 8 cameras, and the PETS dataset [292], EPFL Terrace [293], EPFL-RLC [294], and WILDTRACK [295] synchronize up to 8 cameras for pedestrian multi-camera tracking [293], The latter set provides annotations in a unified ground plane, with pedestrians represented as points [295] or grid cell occupants [294] on the ground plane. In a vehicle context, the CityFlow dataset [296] associates 2D MOT data in a traffic monitoring context across multiple cameras throughout a city, with an average of 4 cameras covering scenes, but object dimensions and a shared tracking space are not modeled. NuScenes contains multiple frontal, side and rear-facing, frame-capture synchronized cameras enabling 3D multiple-camera tracking in an AV context. The pNEUMA Vision dataset [297] provides up to 10 drone-mounted camera views and scenes of up to 13 minutes in duration, though has known annotation shortcomings. Synthehicle [298] contains synthetic 3-minute scenes with up to 7 cameras in a traffic monitoring context, totalling over 17 hours of video footage. Crucially, there is no multi-camera dataset with a high object density (over 100), long object durations (5+ minutes), and more than 25 overlapping cameras. Moreover, to the best of our knowledge, no dataset with multiple overlapping traffic videos from real traffic cameras for tracking with 3D vehicle representations exists. Table 2.3 summarizes the listed datasets.

Dataset	Context	Imagery	ReID	Detection		MOT		MCT		Trajectories
				2D	3D	2D	3D	2D	3D	
WILDTRACK [295]	Pedestrian	✓	✓	✓		✓		✓	✓	
EPFL [293, 294]	Pedestrian	✓	✓	✓		✓		✓	✓	
PETS [292]	Pedestrian	✓	✓	✓		✓		✓		
Duke MCMT [291]	Pedestrian		✓	✓		✓		✓		
MOT [82, 217]	Ped/AV	✓	✓	✓		✓				
KITTI [284]	AV	✓	✓	✓	✓	✓	✓			
NuScenes [285]	AV	✓	✓	✓	✓	✓	✓	✓	✓	
Waymo [124]	AV	✓	✓	✓	✓	✓	✓	✓	✓	
VisDrone [282]	Aerial	✓	✓	✓		✓				
TAO [283]	Various	✓	✓	✓		✓				
NGSIM [49]	Trajectory	✓								✓
HighD/InD [66][72]	Trajectory									✓
BoxCars116k [115]	Traffic	✓	✓	✓	✓					
UA-DETRAC [215]	Traffic	✓	✓	✓		✓				
CityFlow [296]	Traffic	✓	✓	✓		✓		✓		
pNEUMA Vision [297]	Aerial	✓	✓	✓		✓		✓		✓
Synthehicle [298]	Traffic	✓	✓	✓	✓	✓	✓	✓	✓	

Table 2.3: Existing Multiple Object Tracking and Vehicle Trajectory Datasets. *Imagery* indicates that video data is publicly available for the dataset.*MOT* indicates multiple object tracking (in 2D or 3D), *MCT* indicates multiple camera tracking (with 3D tracking requiring a unified space in which objects are tracked), and *Trajectories* indicates that the dataset provides fine-grained data suitable for traffic analysis.

3. The I-24 MOTION Open-Road Testbed

Transportation science is undergoing a digital transformation in which increasingly automated vehicles are being developed and deployed on roadways, changing the fundamental physics of traffic flow. Even a small number of automated vehicles can have a direct impact on the macroscopic behavior of traffic flow, highlighting the need to monitor and observe traffic flows across microscopic and macroscopic scales.

The main challenge in understanding broad system-level properties in mobility such as overall energy efficiency, safety, or flow stability is that these properties depend on the driving characteristics of all vehicles in a traffic stream, and each vehicle must be analyzed to a very fine level of detail. *Trajectory data*, or absolute positions of each vehicle at regular time intervals, is considered the gold standard for data collection. High-quality vehicle trajectory datasets support research on traffic flow theory [299], driver behavior modeling [300], and many other topics [301]. Unfortunately, such datasets are hard to come by and are limited both in length and duration. For this reason, developing reliable, extensive methods for collection of trajectory data is viewed as one of the largest challenges for continuing traffic flow research [299]. Recognizing the impact of freeway trajectory data collection efforts such as NGSIM [49] and HighD [66] (see also Table 3.1), and emerging urban datasets exemplified by pNEUMA [71], and at the same time the limited availability of sources for trajectory data, we embarked on a 5-year effort to instrument a section of freeway that could help enable the next wave of empirical traffic science that depends on abundant trajectory datasets. Figure 3.1 details the rough timeline of conceptualization, design, testing, and construction we laid out to achieve this goal. This chapter presents the outcome of that effort, resulting in an instrument known as I-24 MOTION.

The primary contribution of this chapter is to introduce a new traffic instrument to address the challenge of reliably producing vehicle trajectory data at a large spatial-temporal scale. The Interstate 24 MObility Technology Interstate Observation Network (I-24 MOTION) is a new traffic sensing instrument and testbed to provide continuous, complete vehicle trajectory data in a long-term, ongoing manner. The system consists of 294 4K traffic cameras mounted on 40 traffic mast poles along 4.1 miles of 4-5 lane interstate roadway near Nashville, Tennessee. Cameras are positioned such that every foot of approximately 4.5 miles of roadway is visible in at least one camera field of view. The resulting video data is transferred via a dedicated fiber optic network to a centralized compute cluster for processing and anonymous vehicle trajectory generation. Relative to existing drone-based vehicle trajectory datasets, the task of producing trajectory data from relatively lower infrastructure-mounted cameras poses non-trivial computer vision challenges, most notably significant object occlusion and a requirement for tightly calibrated camera homography information.

The rest of this chapter is organized as follows: Chapter 3.1 lays out the early conception of needs the system was designed to meet. Chapter 3.2 describes the initial design considerations culminating in a one-week, 6-camera test to assess the feasibility of this idea in 2018-2019. Chapter 3.3 describes a scaled-up validation system completed in 2020 comprised of 3 poles and 18 cameras, and the corresponding hardware and software design this system enabled. Chapter 3.4 briefly describes considerations in scaling findings from the validation system to the larger full system. Chapter 3.5 describes the full I-24 MOTION system, completed in November 2022. Lastly, Chapter 3.6 describes some initial data analyses illustrating the utility of the data both in corroborating existing findings and enabling new discoveries.

Dataset	Location	Context	Year	Cameras	Time Scale	Spatial Scale	Vehicles
NGSIM US-101[49]	Los Angeles, CA	5-6 lane highway	2005	8	0.75 hr	0.64 km	9,206
HighD [66]	Cologne, GE	2-3 lane highway	2018	1	16.5 hr	0.42 km	110,500
ExiD [67]	Aachen and Cologne, GE	2-4 lane interchanges	2021	1	16.1 hr	0.42 km	69,172
Automatum [68]	GE	2-4 lane highway	2021	1	30 hr	0.66 km	60,000
HIGH-SIM [69]	I-75, FL	3-4 lane highway	2021	3	2 hr	2.44 km	-
Zen Traffic Dataset [80]	Osaka, JP	2 lane highways	2018	-	5 hr	$\sim$2 km	-
I24-MOTION (released)	Nashville, TN	4-5 lane highway	2022	276	47 hr	6.75 km	$\sim$600,000
I24-MOTION (planned)	Nashville, TN	4-5 lane highway	2023	276	daylight	6.75 km	$\sim$150,000/day

Table 3.1: Comparison of existing highway complete vehicle trajectory datasets. "$\sim$" indicates approximate value. "-" indicates data is not available.

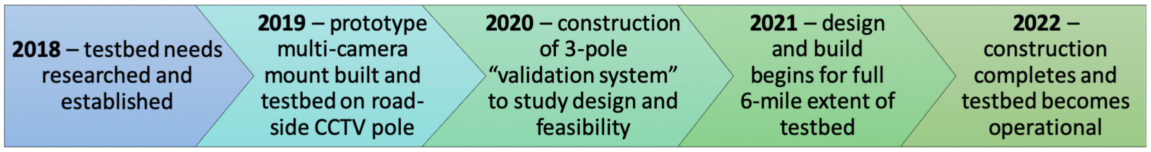

2018 – testbed needs researched and established
2019 – prototype multi-camera mount built and testbed on road-side CCTV pole
2020 – construction of 3-pole “validation system” to study design and feasibility
2021 – design and build begins for full 6-mile extent of testbed
2022 – construction completes and testbed becomes operational

Figure 3.1: Broad timeline of testbed conceptualization, design, and construction.

3.1 Identification of System Needs and Benefits

This section defines the needs that the testbed fulfills laid out in the early project conceptualization phase, related to operations, research, and development. The overall design of the system as a dense network of cameras, arose specifically to meet these needs. The needs met by the testbed span operations, development and research. [1]

Need 1 – Operations: The I-24 corridor is a major limited access facility within Tennessee for commuters and freight. This corridor was selected for the state’s first Integrated Corridor Management (ICM) project, called the I-24 SMART Corridor, which operates on the route between Nashville and Murfreesboro. The corridor includes Interstate 24, the parallel arterial route SR 1, and connector routes between I-24 and SR 1. The ICM project has deployed an upgraded communications network and Intelligent Transportation System (ITS) devices for increased operational management of the corridor. High resolution sensing provided by the testbed in this area will allow TDOT to better leverage the existing ICM infrastructure investments for refining operational strategies beyond the limited-fidelity decision making capabilities using aggregate sensing technologies.

Need 2 – Development: Currently, most test facilities used by industry for vehicle and vehicle technologies development are closed-course environments that enable safe testing of experimental technologies. Due to the intended purpose of the closed course testbeds, it is however very challenging to understand how the proven technologies will operate and interact in real world environments. The variability of traffic conditions and the unique human driver behavior inherent to a real roadway are challenging conditions for new technologies, but necessary barriers to overcome. This project will be a novel testing facility that will allow TDOT and third-parties the ability to gather data on new transportation technologies through real world testing. The capabilities of the testbed provide the unique ability to collect data from every vehicle on the roadway to evaluate direct and indirect effects amongst the entire traffic stream.

Need 3 – Research: Producing vehicle trajectory data allows features of traffic flow related to individual vehicle behavior to be explained. Such data at the level of individual vehicles is more important than ever due to increasing autonomy on individual vehicles, which are beginning to influence traffic flow via their interactions with conventional vehicles. The testbed’s 6-mile length allows for observing complex multi-vehicle interactions such as the creation of phantom traffic jams. The testbed will generate over 200,000,000 vehicle-miles of trajectory data annually, assuming an 80% uptime. The testbed location exhibits a wide range of traffic conditions from free-flow to heavy congestion and bottlenecks.

3.2 6-Camera Feasibility Study

To enable the the above needs, this section describes the initial design considerations and feasibility study carried out to gauge how effective a multi-camera based vehicle tracking system would be for trajectory extraction. [2]

3.2.1 System Architecture

We discuss major decisions and motivating factors in the design of I-24 MOTION with respect to the type and number of sensors deployed, the computing regime (central versus edge), and the processing pipeline

[1]This section is adapted from [85] with permission from authors William Barbour, Meredith Cebelak, Brad Freeze, and Daniel B. Work.

[2]This work was published in and is adapted from [84].

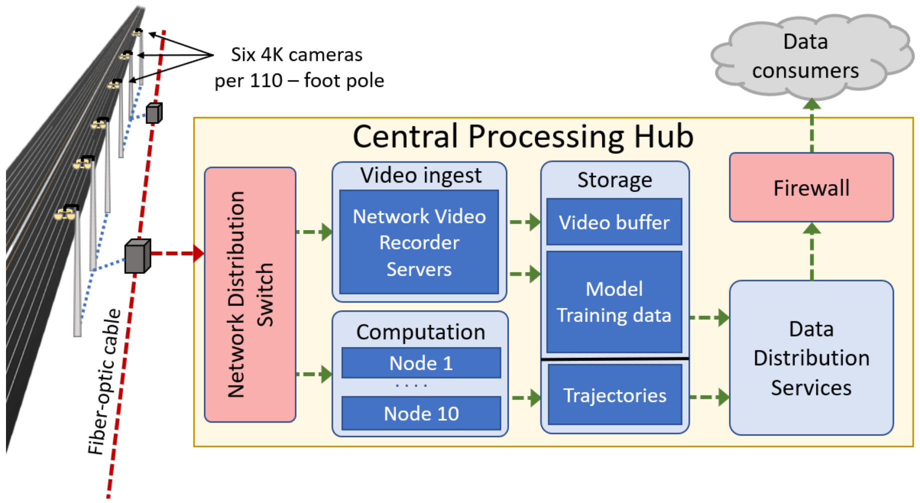


Figure 3.2: I-24 MOTION system overview.

used to convert sensor readings into trajectory data. Figure 3.2 provides a high level overview of the system networking, hardware and storage components in the preliminary design. A video ingest in the central processing hub stores video in a rolling buffer while computation nodes process the data stream in real time.

3.2.1.1 Sensor Components

Sensors for vehicle trajectory data collection were selected according to spatial and temporal resolution as well as ruggedness. Sufficient spatial resolution was required to localize a vehicle within approximately one foot of its absolute position to provide high-quality trajectories. A temporal resolution of 10 Hz was required to capture high-speed changes in traffic conditions such as extreme braking events, as defined based on [49]. Sensors were also required to be rugged enough to withstand normal and severe weather conditions such as heat, ice, and water for an expected operating lifetime of five years. Based on these constraints, two main categories of sensors were considered for I-24 MOTION:

- *Light Detection and Ranging (LIDAR) Scanners* - Laser distance scans are used to capture geometric information about a scene with high resolution. Current models provide readings at 15 Hz and capture information at 0.1 deg intervals.
- *Cameras* - Current 4K resolution sensors capture complete visual information from a scene at 30 Hz and at 0.028 deg intervals (2160 pixels over a 60-degree field of view).

Cameras were ultimately selected for I-24 MOTION. At selection time, LIDAR units were an order of magnitude more expensive than 4K resolution cameras suitable for traffic monitoring. Cameras preserve color and lighting information, which aids in vehicle re-identification and other data analyses besides trajectory extraction; LIDAR does not. Cameras provide higher resolution than current LIDAR models and cover roadway at greater distances with sufficient resolution. Lastly, cameras have been used in traffic operations for over 20 years, so are well-proven as a traffic monitoring solution, whereas LIDAR units are not traditionally used for long-term fixed traffic monitoring installations. Unlike LIDAR, which has good performance in day and night time conditions, a challenge for cameras is that the performance can deteriorate in low lighting conditions.

3.2.1.2 Infrastructure Components

The precise placement of camera poles along the roadway will greatly impact the resolution and completeness of the data collected. Ideally, poles should be tall enough to provide an un-disrupted viewpoint of the roadway. However, logistical and cost considerations limit the pole height to 110 feet, and all poles must be located on one side of the freeway. Thus, four main considerations informed camera placement:

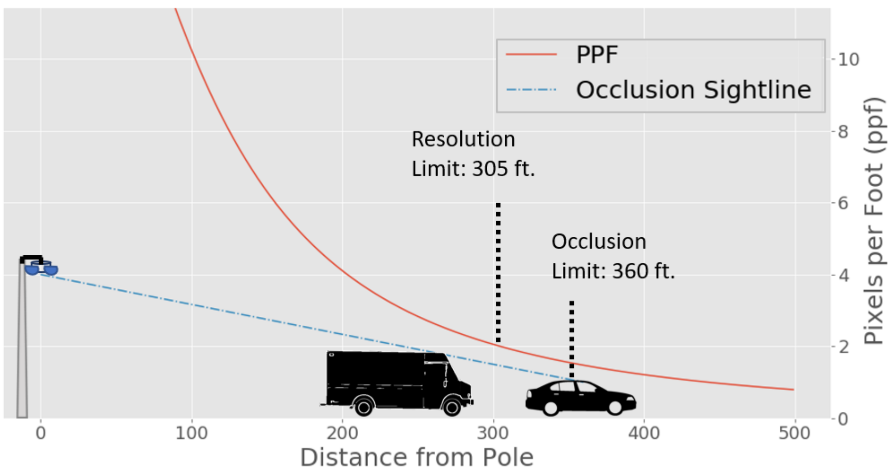


Figure 3.3: Parallel occlusion and resolution limit diagrams.

- *Perpendicular Occlusion* - Vehicles in a lane closer to the camera can *occlude*, or block, vehicles in lanes farther from the camera. This type of occlusion occurs perpendicular to the roadway, in which direction vehicles are spaced approximately 12 feet apart on center in standard width lanes.
- *Parallel Occlusion* - Tall vehicles sufficiently far from the camera can occlude short vehicles travelling in front of them. This type of occlusion happens in sight-lines roughly parallel to the roadway, in which direction vehicles may be closely spaced in slow-moving traffic.
- *Resolution* - Cameras and LIDAR sensors both provide constant angular resolution, but this angular resolution covers an increasingly large distance along the roadway at locations farther from the sensor. To enable both accurate vehicle detection as well as to enable a variety of other use cases, a minimum resolution of 2 pixels per foot along the roadway is required. This means that a section of road from d to $d+1$ feet from the camera must have at least 2 pixels covering it in the direction parallel to the roadway.
- *Field of View* - A sufficient number of cameras must be placed such that their fields of view cover the entire roadway and also overlap.

Though some state-of-the-art object detection algorithms provide sub-pixel accuracy, a larger margin of error was added to account for algorithm inaccuracies and detection difficulties for I-24 MOTION. Camera placements were calculated to provide a minimum 2 pixels per foot along the roadway using a straightforward calculation. Perpendicular and parallel occlusion limits were calculated to determine whether all lanes will be free from perpendicular occlusion and at what distance parallel occlusion will become a limitation. Standard vehicle dimensions and spacings were used, and vehicles less than 50% visible were considered occluded. For calculations, we assumed a de-rated pole height of 100 ft to account for varying terrain elevations adjacent to the roadway. From these constraints, we found that a 4K (3840 x 2160 pixel) resolution camera could provide 2 pixels per foot along the roadway up to 305 feet from the pole. Based on resolution and occlusion constraints (resolution governs as seen in Figure 3.3), a conservative coverage radius of 250 feet was selected for each pole.

Field of view calculations were carried out along the freeway via a 3D model to ensure sufficient cameras are mounted at each location to provide complete coverage of the entire radius of coverage along the roadway. Based on the roadway width for the I-24 MOTION, it was determined that at least five cameras per pole were necessary to provide sufficient coverage of the radius of coverage. Figure 3.4 shows the resulting configuration.

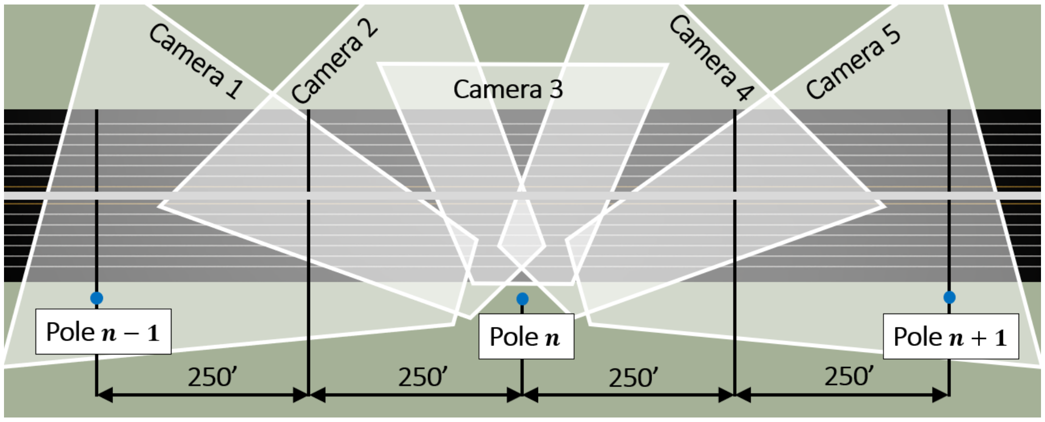


Figure 3.4: Camera field of view alignments for complete roadway coverage based on preliminary design calculations.

3.2.1.3 Computing Regime

To extract trajectories from the video data, we considered both a centralized processing approach and an edge computing approach. Recently, edge computing has been a popular choice for IoT mobility sensor applications [302, 303]. I-24 MOTION will have roughly 400 4K cameras when completed. Using H.264 compression, a conservative estimate for the network bandwidth requirement is 15 gigabits per second (GBps). Edge processing reduces network bandwidth requirements, which can be favorable if the network bandwidth is limited (for example on a cellular network). If raw data contains *personally identifiable information* (PII), edge computing can be used to strip the data of PII. Moreover, decentralized approaches can improve system robustness by eliminating single points of failure of the computing resources.

Despite these advantages, most edge computing solutions are limited in terms of *graphics processing unit* (GPU) computation performance compared to centralized computing solutions. The object detection algorithms in I-24 MOTION will ideally be near real-time speed (at the rate at which data is produced by cameras). It is possible that specialized edge compute resources can maintain high frame-rates at lower resolutions and when there are few objects in the frame [302], or by intelligently sharing on and off-edge compute resources as in [304]. Thus, in I-24 MOTION we adopted a centralized computing paradigm. Moreover, co-locating all of the computational resources allows us to dynamically reallocate compute resources to manage workload and scale the computing needs as more algorithms are deployed to extract additional information from the videos. I-24 MOTION is located concurrently with a pre-existing optical fiber network installed by the Tennessee Department of Transportation for intelligent transportation system applications which supports 40 GBps of traffic, a suitable private network for data transfer.

3.2.2 Software Components

The software processing pipeline must reliably and accurately convert camera data into vehicle location data. Object detection, tracking, and trajectory conversion algorithms must run in real-time with respect to the input rate to enable the camera network to operate continuously for a long-term deployment. We assume that small inaccuracies and fluctuations in readings, as well as temporary losses of a vehicle's location, can be smoothed and corrected in post-processing steps.

3.2.2.1 Vehicle Detection Algorithm

For I-24 MOTION, *object detection* algorithms are used to extract vehicle positions from camera image data. The task of object detection is well explored, and mature algorithms exist for efficiently detecting objects. For example, YOLO-v3 [130] and Faster R-CNN [156], are two state of the art object detection algorithms. Both algorithms are based on the use of *convolutional neural networks* for extraction of high-level information from image pixels; the main difference is that Faster-RCNN relies on a two-stage detection framework in which rough detections are first output, and then this set of detections is used to create a second, more refined set of final predicted object locations. YOLO, on the other hand, relies on a single prediction stage, making it faster but slightly less accurate. In this work a pre-trained YOLO-v3 model was used for object detection.

3.2.2.2 Tracking

Object tracking is the task of locating the same objects in consecutive frames of video data. For I-24 MOTION, a *tracking by detection* approach is employed, in which object detection is performed on every frame, and subsequently detected objects are matched between frames. Several accurate algorithms for object tracking exist based on modeling object types and behavior [305], filtering [5], and direct output from neural networks [229]. The *Simple Online Realtime Tracking* (SORT) algorithm [5] was used for tracking for its low computational overhead and high accuracy. It uses a Kalman filter to predict and correct the positions of each vehicle over time [306].

To implement SORT, the state of each vehicle:

$$\mathbf{x}_n = [x_n, y_n, s_n, r_n, \dot{x}_n, \dot{y}_n, \dot{s}_n]^T, \tag{3.1}$$

is expressed as a 7-dimensional state vector where x and y denote the bounding box center coordinates, s is the width of the bounding box, r is the width-to-height ratio of the bounding box, and $\dot{x}$, $\dot{y}$, and $\dot{s}$ denote the rate of change of x and y. A constant velocity model is assumed resulting in a state space model (presented here for a single vehicle) of the form:

$$\mathbf{x}_{n+1} = \mathbf{F}\mathbf{x}_n + w_n\,, \quad \mathbf{y}_n = \mathbf{H}\mathbf{x}_n + v_n, \tag{3.2}$$

where $\mathbf{x}_n$ denotes the state at timestep n, $w_n \sim \mathcal{N}(0, \Sigma_w)$ is the process noise, $\mathbf{y}_n$ is the measurement at timestep n and $v_n \sim \mathcal{N}(0, \Sigma_v)$ is the measurement noise. The dynamical model $\mathbf{F}$ and the observation model H are written explicitly as:

$$\mathbf{F} = \begin{bmatrix} 0&0&0&0&\Delta t&0&0 \\ 0&0&0&0&0&\Delta t&0 \\ 0&0&0&0&0&0&\Delta t \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{bmatrix} \tag{3.3}$$

where Δt is the time between consecutive video frames. The system is observable and thus can be accurately estimated with a Kalman filter according to the update and measurement equations 3.2. Other state spaces and model dynamics were also considered but were found to perform worse.

An important detail in the tracking problem is the assignment of detected objects to the correct vehicle in the state space. This is done by matching the predicted positions in the model prediction step of the Kalman filter to the actual detected objects from the object detector, using the Hungarian algorithm for bipartite matching [307]. Once the assignment is known, a standard Kalman update can be performed to correct the predicted state based on the measurement. Figure 3.5 shows the use of this method for tracking object trajectories through consecutive frames of video from a test of I-24 MOTION.

3.2.2.3 Trajectory Conversion

To be useful for intelligent mobility applications, tracked object trajectories must be expressed in absolute coordinates, rather that image space coordinates. Assuming that the ground plane is flat, there exists a *perspective transform* expressible as a 3x3 *homography matrix* that maps points from the image plane to the ground plane while preserving straight lines. If four points in image space and their corresponding ground plane points are known, a straightforward system of linear equations can be solved to determine the 8 parameters of the transform $a_{11}, \cdots, a_{32}$ (by convention the last parameter is always 1). Then, an arbitrary image plane point (x_n, y_n) can be mapped to its corresponding ground plane point (x', y') via:

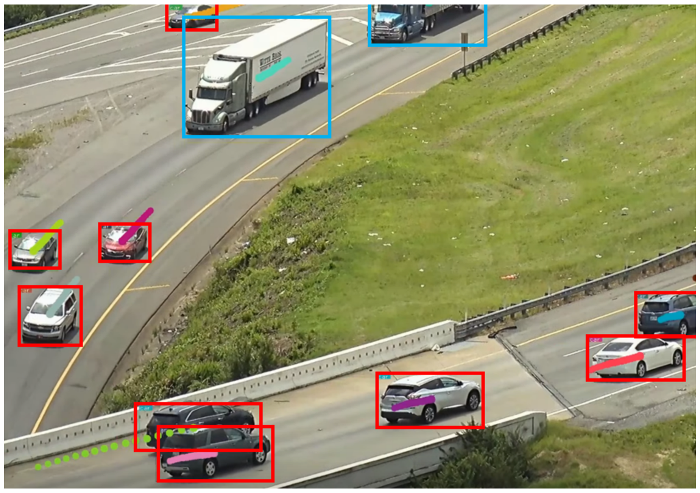


Figure 3.5: Vehicles tracked with YOLO and SORT. Red boxes denote cars, Blue boxes denote trucks. Point trails denote the position of the associated vehicle in prior frames.

$$\begin{bmatrix} i \\ j \\ k \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.4}$$

$$x' = i/k \text{ and } y' = j/k, \tag{3.5}$$

where k denotes a scaling coefficient.

3.2.3 System Resilience

I-24 MOTION provides robustness to hardware and software failures by way of redundancy:

- *Single Camera Failure* - Cameras are pan-tilt-zoom enabled and six cameras are mounted per pole for redundancy. In case of a camera failure, five cameras can be re-positioned to seamlessly cover the area of observation.
- *Single Pole (Networking Hardware) Failure* - Cameras on neighboring poles have overlapping fields of view and can cover the area of the failed pole (with possible occlusion).
- *Single Compute Node Failure* - Load balancing can redistribute computational load to other compute nodes. If available compute resources cannot keep up with data influx, the frame resolution can be reduced to speed up the computationally expensive object detection step.
- *Storage Failure* - Data is stored at multiple locations and can be restored to failed location after the failure is addressed.

3.2.4 Feasibility experiments

Preliminary experiments were carried out to verify the feasibility of the proposed sensors, physical infrastructure, and computational pipeline. Data was collected from a single six-camera pole for one week from August 9-16, 2019. This data was then processed with the pipeline described above to produce a trajectory dataset. To enable tests with six cameras per pole, a custom mounting bracket and associated networking hardware was also designed, prototyped, and tested (Figure 3.6). This prototype serveed as a feasibility analysis for larger-scale deployment of such a multi-camera mount. Code and videos from the test can be found at http://github.com/DerekGloudemans/I24-MOTION-examples. Figure 3.7 provides an overview of the feasibility experiment.

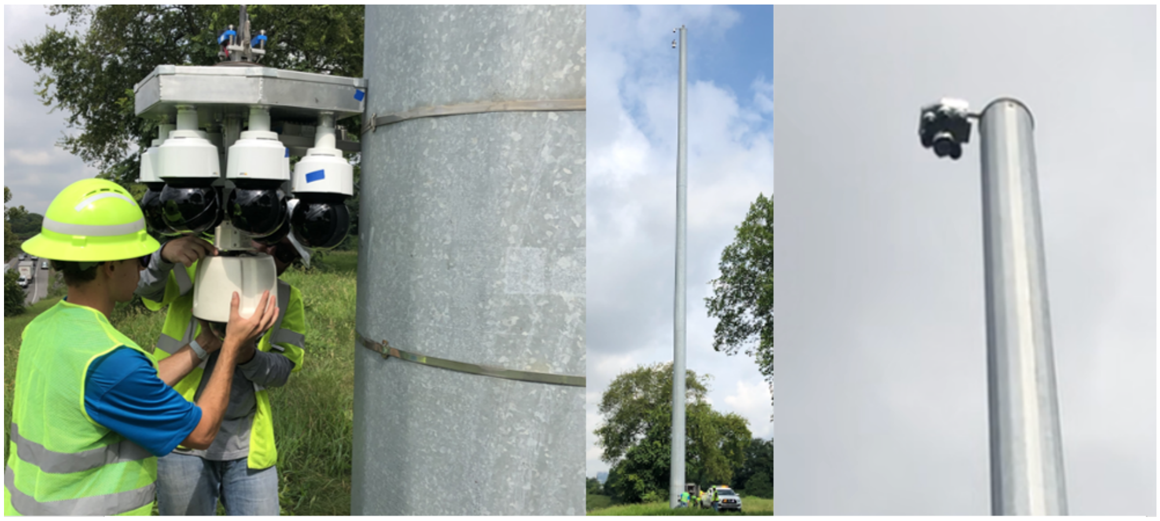


Figure 3.6: Multi-camera mount is raised onto 110 ft pole.

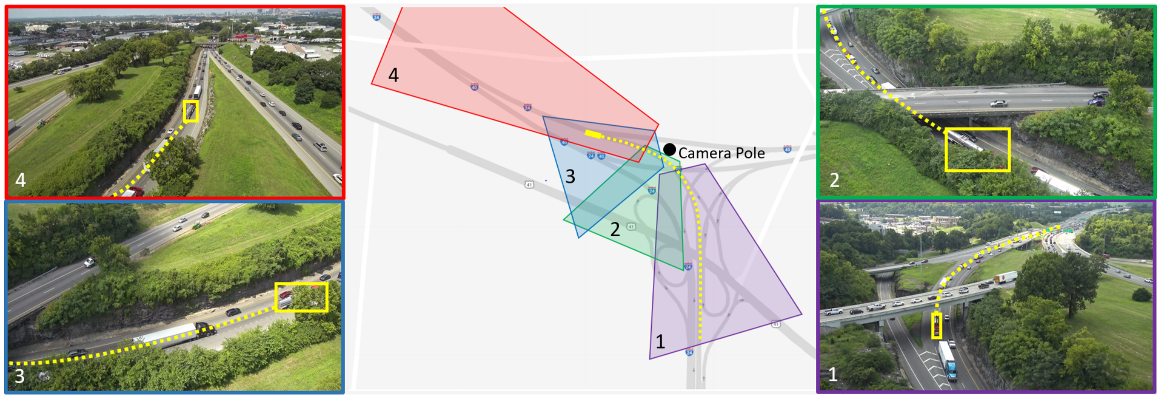


Figure 3.7: I-24 MOTION feasibility test installation. Multiple overlapping 4K cameras enable tracking vehicles seamlessly along the roadway.Of the six cameras mounted on the pole, four are shown here for simplicity.

3.3 18 Camera Camera Validation System

Based on the success of the feasibility experiment, in collaboration with the Tennessee Department of Transportation, a three-pole system was constructed that demonstrated the concept and design of the system in operation. This section describes the validation system hardware and design decisions made for the testbed. Chapter 3.4 lays out additional considerations and plans for expanding from the current validation system to the full testbed build out. [3]

As discussed in the previous section, camera-based sensing is advantageous for observing the entire roadway; and a dense deployment of cameras, such that their views overlap and observe vehicles continuously, is necessary for providing end-to-end coverage through the testbed. Installing this infrastructure in a permanent capacity provides the temporal coverage of data that is needed for many research applications, and provides a location that is always available for testing technologies or running experiments. For instance, multiple camera poles allows for experiments into the efficacy of tracking objects across cameras and maintaining tight camera time synchronization using existing approaches.

In 2020, TDOT and partners set out to study the feasibility of this dense camera infrastructure approach

[3]This section is adapted from [85] with permission from William Barbour, Brad Freeze, Meredith Cebelak, and Daniel B. Work.

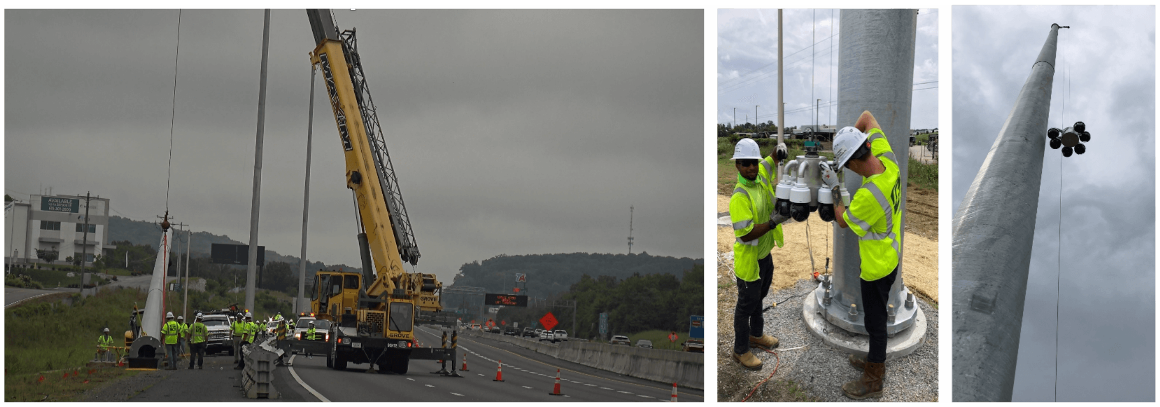


Figure 3.8: Validation system construction. Left: 110-foot poles are raised adjacent to the freeway; middle: Multi-camera mount holds 6 4K PTZ cameras; right: cameras are elevated to the top of the pole.

on I-24. A 3-pole, 18-camera system was designed, constructed, and commissioned (Figure 3.9, Figure 3.8). As is intended with the full testbed, cameras were connected via fiber to a hub building that houses servers for real time trajectory extraction. The prototype was strategically sized to inform meaningful scalability and design considerations, thereby determining if the full testbed scope was attainable with the same design strategy.

The following describe design elements of the 3-pole validation system, effectively allowing us to pilot each technology and strategy for inclusion in the full system.

- **Pole & foundation design.** The validation system has cameras mounted on 110-foot tall poles (Figure 3.8 Left). To observe all vehicles on the roadway with minimize occlusion, the poles are significantly taller than the standard 30-50 ft poles used on many other CCTV systems. New poles and corresponding foundations were designed and built with total deflection of less than 1.5 inches in 30 mph wind.

- **Fiber network.** The poles are connected on a fiber backbone to a hub building capable of hosting compute and storage equipment. As part of a larger infrastructure and technology upgrade by TDOT known as the I-24 SMART Corridor, the freeway fiber network is complete and has dark fiber available for use by the testbed.

- **Camera lowering device.** The camera lowering device is a critical component of all traffic monitoring cameras in Tennessee (Figure 3.8 right). It allows the camera cluster to be safely lowered to the ground for routine cleaning and maintenance using a winch at the base of the pole. The validation system confirmed that the new-generation lowering device is also able to transmit data from six 4K resolution video cameras through the lowering device and down the pole where it is tied into the existing TDOT fiber network.

- **Cameras and camera mounting bracket.** The system uses a custom, 6-camera mount (Figure 3.8 middle) attached to a lowering device. Camera pan/tile/zoom capabilities allow remote alignment to achieve the necessary 180-degree overlapping field of view across cameras on each pole and between camera poles. Deploying multiple cameras to each pole extends coverage of the testbed by reducing the number of poles needed. The weather tight camera mount holds a network switch responsible for making data streams transmittable through the lowering device.

- **Hub building access.** The landing point for video data from the camera network is the TDOT network hub building. At this point the fiber optic cables are connected to a network switch that connects the computational and data servers responsible for buffering video data, computing vehicle trajectories, and storing resultant data.

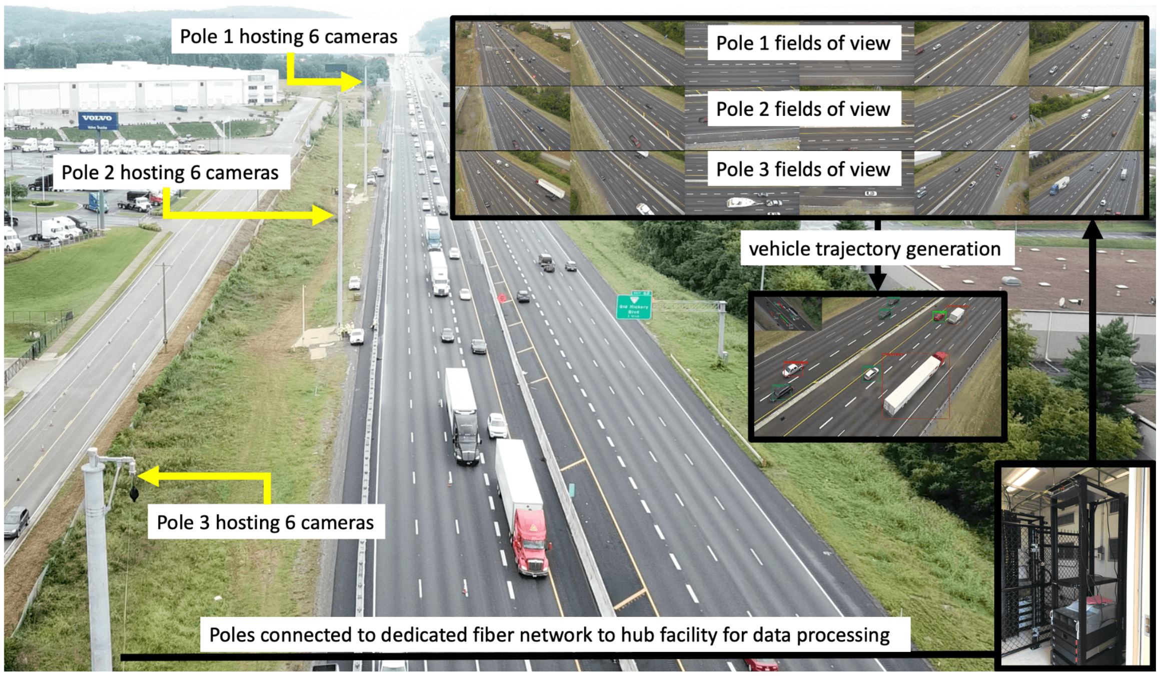
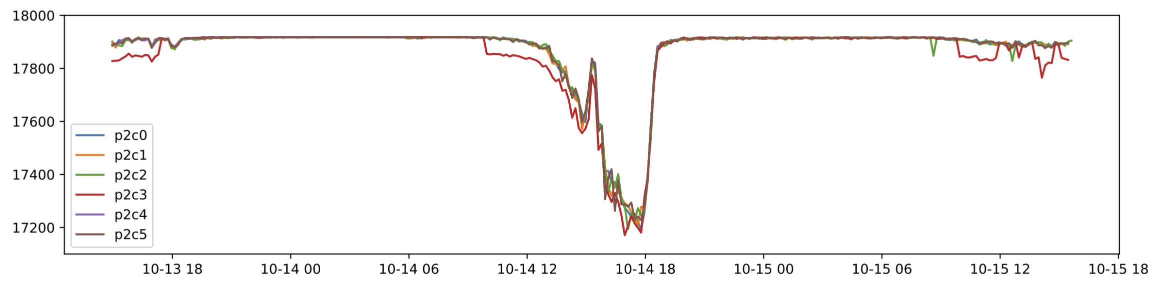
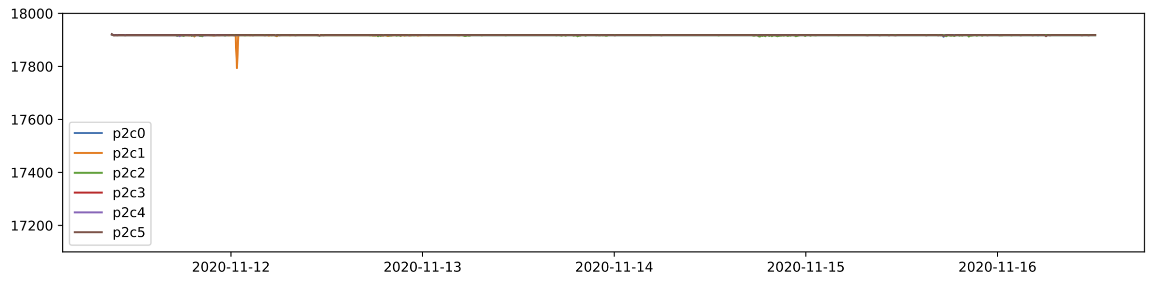


Figure 3.9: Testbed overview. Poles spaced approximately 500-600 ft host multiple 4K resolution cameras to capture an overlapping and continuous field of view of the roadway. Video is streamed to a hub facility for data processing, where the video is converted to trajectory data for analysis.

- **Video ingestion.** Camera streams are mapped to individual servers tasked with handling the raw network stream and producing decoded video frames for processing. This ingestion required a custom data pipeline to be written in GStreamer, which the team completed in consultation with experts at Nvidia and RidgeRun. The resulting pipeline can dynamically route and buffer video streams, archive video snapshots of interesting segments, and monitor the integrity of incoming video. Each camera clock is set by a centralized network time server. Synchronizing at camera capture time rather than server receive time allows precise synchronization across cameras. The pipeline serves as the interface point at which computer vision tracking algorithms take decoded video frames as input for processing.
- **Trajectory generation.** The core of trajectory processing is a custom multi-object tracking algorithm called *crop-based tracking* (CBT) [92] (see Chapter 4). The algorithm is based on existing open-source components – a deep neural network image detector [60, 132] and the *Kalman Intersection Over Union* (K-IOU) tracker [166, 214] – along with enhancements we have made for increasing speed and accuracy. The primary improvement in CBT over existing K-IOU is that it exploits the motion model of the Kalman filter to replace computationally expensive detection steps with a cheaper vehicle localizer. The Kalman filter makes a prediction of the vehicle location in the next frame based on its most recent position and the vehicle's individual motion model. The vehicle localizer detects the position of the vehicle within a small region extracted from the full image that is centered around the Kalman filter predicted location. In this manner, a smaller portion of each image is ultimately processed compared to running detection on the entire image. Particularly in sparse or free-flow traffic, most of the image space will not contain vehicles so CBT can be very efficient. The results enable faster tracking without sacrificing accuracy: at least a 250% speedup over tracking by detection using K-IOU, compared on a single offline camera video stream using an Nvidia RTX6000 GPU [91]. We retrained the object detector on 10,000 vehicles to improve performance given the non-standard overhead field of view. Synchronized overlapping cameras enable trajectory stitching in 3D coordinates across the instrument.
- **Uptime.** Frame delivery statistics are tracked from each camera to the servers hosted in the hub building. Initially the system delivered 95% of frames from cameras. This frame loss is shown for six of the



(a) Early reliability issue with video frame delivery.



(b) Frame delivery issue fixed with hardware change.

Figure 3.10: Video frames delivered per 10-minute interval, tracked over time for uptime statistics. (Credit: William Barbour.)

cameras in Figure 3.10; the number of frames delivered across ten-minute intervals is tracked for each camera over 48 hours. Cameras exhibited a simultaneous and dramatic decrease in number of frames delivered – degrading down to a 4% loss rate in this particular case. It was discovered that an upgraded ethernet surge arrester was needed for this application and the upgrade resulted in 99.95% of all frames received; the frame delivery tracking is shown after the upgrade in Figure 3.10. Frame counts now exhibit very little variability, save for a small momentary drop experienced by one of the cameras.

3.4 Full System Build-out

Design and construction of the full-length testbed primarily replicated the successful strategy of the validation system. Modifications to accommodate the larger scale focused mostly on the cyber infrastructure. The detailed timeline for full system design, construction, and commissioning phases is given in Table 1; completion of construction is scheduled for Q3 2022, with all functions of the fully-commissioned testbed scheduled to be online in Q2 2023. We detail the major considerations for the full system here.

Location

The physical assets of the testbed (e.g., poles, fiber, cameras) are located on I-24 between the interchanges of Bell Rd. (approximate westernmost extent), to Waldron Rd. (easternmost extent); see Figure 3.11. The stretch includes a major recurring bottleneck, approximately 13% truck traffic, and more than 150,000 vehicles a day that pass through the roadway. This will effectively extend the testbed in the direction of Nashville from the site of the validation system. Placement of camera poles in the testbed area will continue the strategy of minimizing occlusion by using the same 110 ft tall poles, placing them directly adjacent to the roadway shoulder, and spacing poles 500-600 ft apart.

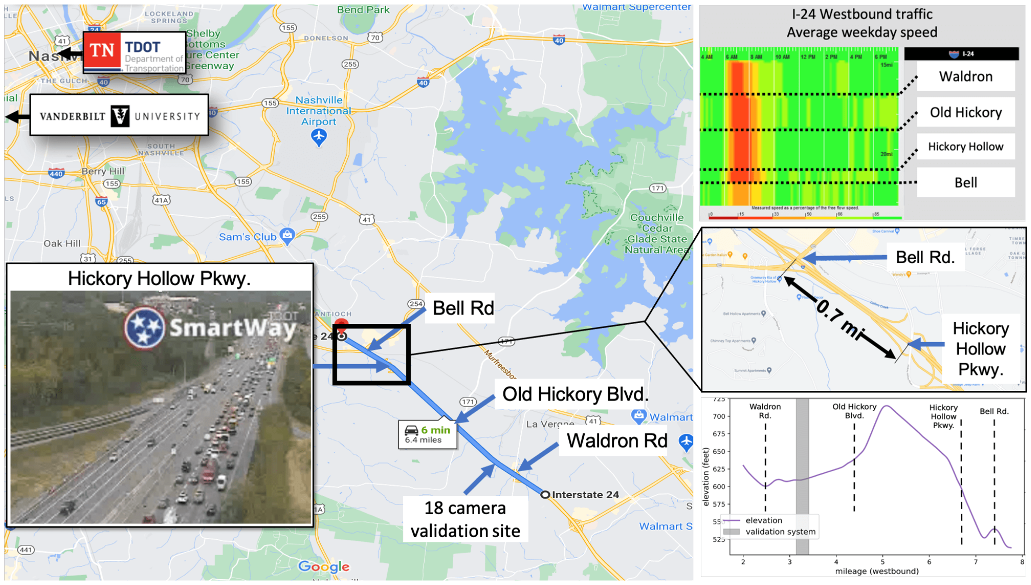


Figure 3.11: Overview of the selected full testbed location in Nashville, TN. (Credit: William Barbour.)

Cameras, poles, and physical infrastructure

Cameras, poles, network switches, lowering devices, etc., will follow the same design as the 18-camera validation system, including only minor changes that were informed during the construction and testing process. For example a wider field of view camera may eliminate the need for one camera per pole.

Computational infrastructure and networking

The video data streams from the camera network require dedicated hardware and software to ingest, process, and store the large volumes of data. Computational servers are built around graphics processing units (GPUs), which are efficient processors of neural network computations underlying trajectory generation algorithms. Each compute server hosts eight GPUs and the number of required servers is based on tracking algorithm performance on the validation system. This performance will dictate the ultimate size of the computational array for the full testbed.

Processed vehicle trajectory data resulting from the camera network will be made available to the research community by bundling and hosting in a data repository. Trajectory data will be segmented temporally, with recent data available in small segments and older data available as compressed bulk downloads. A data hosting server will automate this rolling process as new data is constantly generated, and also host a dashboard informing users of the data and system status.

Table 3.2: Full testbed design, construction, and commissioning timeline.

	Cyber infrastructure	**Physical infrastructure**	**Project management**
Design Phase			
Q3 2021	Finalize software and hardware architecture.	Locate 1.5 mi. instrument site. ROW plans.	Form executive board for instrument dev. and ops.
Q4 2021	Develop server specifications and place order.	Develop and submit construction plans.	Executive board reviews plans with stakeholders.
Q1 2022	Server hardware, OS, and network configuration.	Construction field review; bid documents and letting.	Develop business and long term funding plans.
Construction Phase			
Q2 2022	Finalize trajectory extraction software package.	Begin construction (major component sourcing).	Write data dissemination and privacy policies.
Q3 2022	Software integration and unit testing.	Construction continues.	Write experimental plan and safety procedures.
Commissioning Phase			
Q4 2022	Full system testing of new construction.	Construction completion, inspection, integration.	Develop infrastructure maintenance schedule.
Q1 2023	Burn-in testing. Preparation of first dataset.	Burn-in testing, reporting, constr. contract close-out.	Review results of tests for contract release.
Q2 2023	Implement automated data hosting and dissemination.		Review validation results of vehicle trajectories.

3.5 The I-24 MOTION Testbed

This section describes the final Interstate 24 MObility Technology Interstate Observation Network (I-24 MOTION), a camera-based trajectory generation system located on I-24 near Nashville, TN. The instrument consists of 276 4K resolution video cameras mounted on 40 poles ranging from 110 ft to 135 ft above the freeway. The cameras are positioned with overlapping fields of view and are connected by a fiber optic network to a compute facility where the videos are converted to vehicle trajectories. The instrument captures approximately 230 million vehicle-miles of travel annually, and experiences regular recurring congestion. [4]

Figure 3.12 illustrates the data captured by I-24 MOTION, showing a time-space diagram spanning 4.2 miles of I-24 westbound traffic during 4 hours of morning congestion starting at 6:00AM. The image is created by plotting all westbound vehicle trajectories and color-coding the points based on the speed of the vehicle. Vehicle lengths, widths, heights, and lateral positions are also measured but not shown. The waves visible in the image propagate at approximately 12-13 miles per hour.

The main contribution of this section is the creation of the I-24 MOTION instrument, which generates large-scale vehicle trajectory datasets. The section provides the description of key elements of the instrument, including the road network geometry and features, the features of the cyber-physical assets that compose the instrument, and the general data processing steps. These elements are critical to understand the uses and limitations of the current and future datasets. For example, as we explain in Chapter 3.5, the cameras are pole-mounted. The height of the poles are selected to minimize occlusion (excellent for generating accurate vehicle trajectories), but the height can allow sway in strong winds (bad for generating accurate vehicle trajectories). Thus, the physical design directly influences the types of artifacts that can be introduced. The datasets released by I-24 MOTION will be provisioned with a digital object identifier and change logs as new data processing algorithms are deployed and as artifacts are removed.

We also provide a preliminary description of the datasets, the known artifacts today, and our plans to improve them over time. It is clear that at a macroscopic scale, the data in the initial release can already

[4]This section is adapted from [34].

support novel macroscopic analysis and insight, since no interpolation is required - all 4 miles are observed. At the same time, we describe known issues (e.g., fragmented trajectories due to tracking failures; fragmented trajectories due to a vehicle crash which damaged hardware on one pole, etc.). Some of these issues will be resolved through instrument maintenance cycles; while others will be resolved with the advancement of better automated data generation methods. As individual datasets mature, and new datasets are introduced, this article will serve as the reference point for users of all future datasets generated by the instrument.

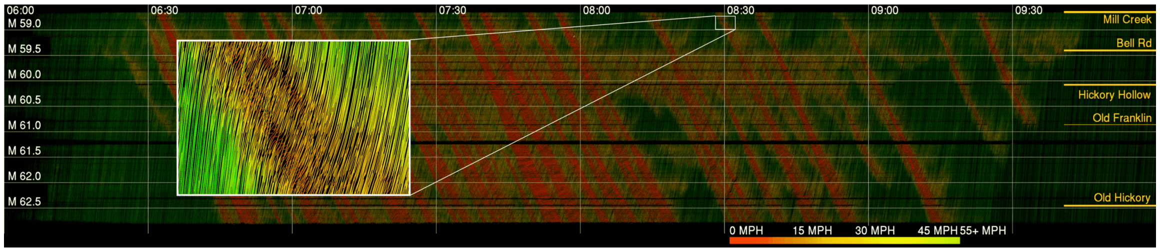


Figure 3.12: Time-space diagram for four hours of I-24 W morning rush hour traffic on Nov 25, 2022, generated from I-24 MOTION vehicle trajectories. x-axis: time of day (HH:MM); y-axis roadway postmile (mi). Postmile decreases for travelers in the westbound direction. A typical congestion pattern is shown with frequent oscillatory traffic observed; and recurring waves travel upstream relative to the direction of traffic at 12-13 mph. The names of interchanges and overpasses appear on the right. The figure inset shows a zoomed in portion of the data which is 0.25 mi in length and 4 min in duration. (Credit: Gergely Zachár and Derek Gloudemans.)

The remainder of this section describes the I-24 MOTION instrument, detailing the physical infrastructure, network and compute hardware, and core algorithms required to provide accurate and complete vehicle trajectory data across a large spatial and temporal scale. The system is still in active development, and continual improvements to improve the reliability, accuracy, and processing speed of the system will be made over the following years. (Chapter 3.6 describes the data produced in more detail, including the recorded quantities, coordinate system, and a comparison in spatio-temporal scale to existing vehicle trajectory dataset, and also provides some preliminary analysis of the data including a characterization of the wave propagation speeds observed in the datasets.)

3.5.1 Physical Infrastructure

The I-24 MOTION instrument provides a continuous field of view of 4.2 miles (6.75 km) on the 4-5 lanes (each direction) I-24 freeway, southeast of Nashville, Tennessee, USA. Pole mounted cameras are connected via a fiber network to a data center, as shown in Figure 3.13, where computer vision tracking and trajectory processing takes place. A total of 276 4K resolution cameras are mounted on 40 poles, each 110-135 feet tall, spaced every 500-600 feet along the freeway. Thirty-four of the 40 poles house 6 cameras each, while 6 poles adjacent to interchanges instead house 12 cameras to provide expanded coverage. The poles provide an overhead vantage point of the road to reduce occlusion, and to provide overlapping fields of view. (A separate 3-pole, 18 camera validation system [85] is located about 0.75 miles eastbound on I-24 from the primary instrument and was used for technology testing and system planning).

3.5.1.1 Location

The location for I-24 MOTION was selected based on traffic conditions, constructability factors, and co-location with other *Tennessee Department of Transportation* (TDOT) initiatives. The four mile section of Interstate 24 is located ten miles southeast of Downtown Nashville and exhibits an *annual average daily traffic* (AADT) of approximately 150,000 vehicles per day across its length [308]. Morning and afternoon rush hour traffic exhibits reliably heavy congestion in opposite directions, frequently reaching stop-and-go conditions, with easily-observable traffic waves on a typical day. I-24 near Nashville is a heavy commuter and freight corridor (10-15% of the vehicle traffic are heavy trucks): it links smaller cities of Murfreesboro, La Vergne, and Smryna with Nashville, and serves as a major shipping and industrial transportation route for Middle Tennessee and the southeast United States.

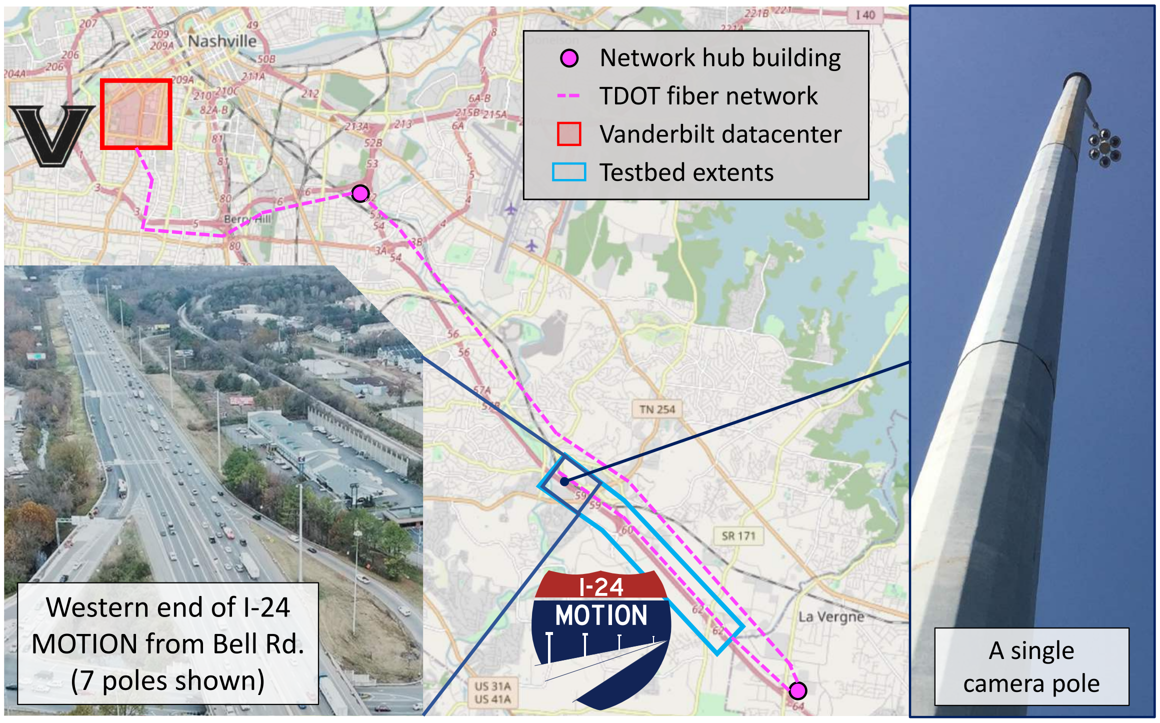


Figure 3.13: Overview of I-24 MOTION site showing location relative to Nashville, TN. The major TDOT fiber network elements and their connection to Vanderbilt University, which houses the trajectory generation algorithms that operate on the live video feeds, are also shown on the map. (Credit: William Barbour.)

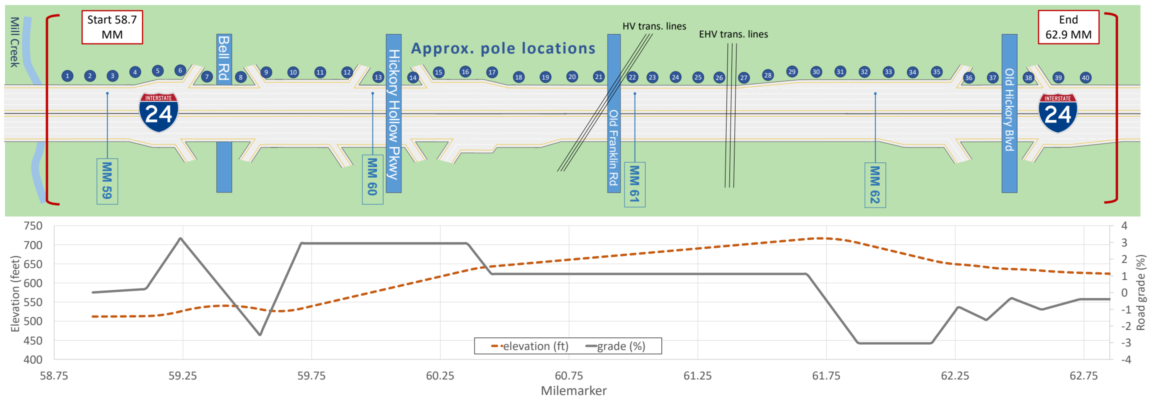


Figure 3.14: **TOP:** Diagram of the I-24 MOTION instrument, spanning from Mill Creek (postmile 58.8) to milemarker 62.8. Camera poles (blue circles) are spaced at roughly 550-foot intervals and are shown in the image relative to other key infrastructure elements. The system spans three overpasses and one underpass, as well as three interchanges with 13 entrance/exit ramps. Relative positions of all elements are correct but diagram is not drawn to scale. **Bottom:** Elevation and road grade along the freeway. Grade is measured in the eastbound (diagram left to right) direction and was determined by differentiating the best-fit second-order piecewise polynomial elevation function. (Credit: William Barbour.)

There are three highway interchanges in this section of I-24: Bell Road, Hickory Hollow Parkway, and Old Hickory Boulevard. Figure 3.14 shows these three interchanges, their on/off ramps, and lane configuration; it also shows the placement of the forty poles with respect to these interchanges and other notable features on the roadway. The roadway elevation and grade is imposed below the roadway diagram in Figure 3.14. The elevation and grade data was processed from aerial-based lidar measurements, conducted and published by the State of Tennessee (`https://lidar.tn.gov`). The processing of this data involved finding the best-fit piecewise second-order polynomial for the elevation data, then differentiating this polynomial to attain the piecewise linear grade function. This approach is consistent with the road design constraint for this roadway of allowing only constant or linear grade in the roadway profile. The elevation and grade data is included as metadata alongside the trajectory dataset.

This section of the I-24 corridor was also selected for the state's first *Integrated Corridor Management* (ICM) project, called the I-24 SMART Corridor, which operates on the 28-mile route between Nashville and Murfreesboro. The ICM project includes Interstate 24, the parallel arterial route SR 1, and connector routes between I-24 and SR 1. The ICM project has deployed an upgraded communications network and Intelligent Transportation System (ITS) devices, such as variable speed limit control, lane control, and ramp metering, for increased operational management of the corridor. This collocation will eventually allow the study of a variety of implemented ITS solutions associated with the I-24 SMART Corridor using I-24 MOTION [309, 310], when the active traffic management systems are enabled.

3.5.1.2 Camera poles

The 40 I-24 MOTION camera poles are each composed of a steel pole structure, ground-level communications and power cabinet, camera lowering device at the top of the pole, and custom camera cluster assembly, each detailed below. Figure 3.15 shows select system components. Camera pole locations, as well as various other landmarks of interest, are included in Appendix B.

As discussed earlier in this chapter,The camera pole system was prototyped across three years at existing pole locations on the TDOT network and with a purpose-built three-pole validation system constructed in 2020 [84, 85]. Valuable lessons from the tvalidation system regarding camera selection, camera cluster mounting position, and pole-to-pole spacing were incorporated in the full system design. The details of the pole components are as follows:

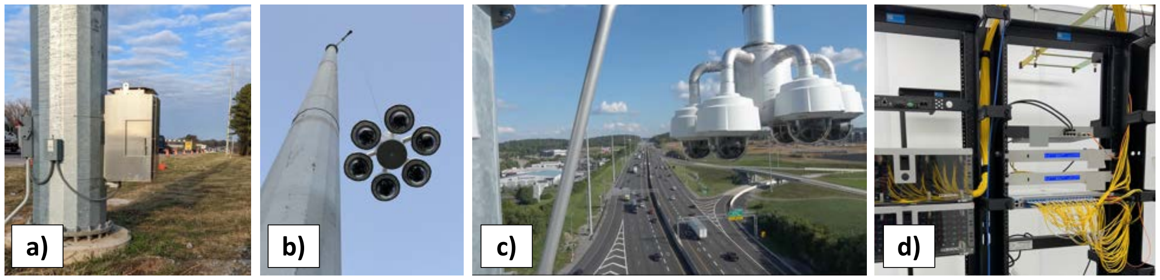


Figure 3.15: Testbed components: a) view of a camera pole base showing the electrical disconnect, transformer, and ground cabinet; b) camera cluster in the process of lowering to the ground with the CLD; c) view of camera cluster at the top of a pole; d) fiber optic junction at network hub building and GNSS network time servers.

- **Steel pole structure:** To observe all vehicles on the roadway with minimal occlusion, the poles are significantly taller than standard 30-50 ft poles used on many other CCTV systems. New poles and corresponding foundations were designed and built to a standard that the total deflection at the top of the pole is less than 1.5 inches in a 30 mph wind. Average pole-to-pole spacing is 550 feet across the instrument, with a minimum of 425ft and a maximum of 625ft due to roadside obstacles and entrance/exit ramps.

- **Ground-level cabinet:** A pole-mounted cabinet (shown in Figure 3.15a) houses a network switch for the fiber optic network, a fiber patch panel, and two power supplies. Power supplies in the cabinet provide DC power to the fiber network switch in the cabinet (65W supply) and to the camera cluster at the top of the pole (240W supply). A fiber communications backbone is present throughout the instrument and links each pole to a communications hub building (shown in Figure 3.15d) and the rest of the TDOT network. Each pole maintains a one gigabit per second network link to an aggregation network switch in a star network topology. This network topology helps simplify configuration and troubleshooting and has additional resilience in the case of some physical damage scenarios.

- **Camera lowering device:** The *camera lowering device* (CLD) is a critical component of all traffic monitoring cameras in the instrument. It allows the camera cluster to be safely lowered to the ground (see Figure 3.15b) for routine cleaning and maintenance using a winch at the base of the pole. While typically configured for only a single camera on a CLD, manufacturer collaboration and internal bench testing confirmed that the lowering device could support simultaneous data transmission from six 4K resolution video cameras to the ground-level cabinet where it ties into the fiber network. The CLD also contains redundant ethernet and power connections that can be utilized without the need for physical access to the top of the pole in case of a connector failure. The CLD is mounted to the top of the pole with a 54-inch extension arm and angled support strut (shown in Figure 3.15c) for added rigidity.

- **Camera cluster assembly:** Mounted on each pole is a custom, 6-camera mount attached to the camera lowering device (shown in Figure 3.15c). The orientation of the camera cluster is orthogonal to the roadway direction(s) of travel. The weather-tight camera mount holds a network switch which aggregates six video data streams to transmit them through a single gigabit ethernet connection on the CLD. The network switch receives DC power from the pole cabinet and supplies power over ethernet (PoE+) to each of the six cameras at 25.5W. On the six poles adjacent to the three interchanges within the instrument, a second camera cluster assembly is mounted in an orientation pointing towards the under/overpass; in the future these cameras will support trajectory generation for vehicles as they enter and exit the highway.

3.5.1.3 Video cameras

The cameras on the instrument are a 4K resolution pan/tilt/zoom (PTZ) network IP model, powered by power over ethernet. The PTZ capabilities allow remote alignment to achieve the necessary 180-degree overlapping field of view across cameras on each pole, as seen in Figure 3.16, and between camera poles. Deploying multiple cameras to each pole extends coverage of the instrument and reduces the number of poles needed. While cameras with wider image field of view exist, these suffered in testing from distortion at the edges of the image that could not easily be corrected to the accuracy needed for coordinate localization.

A critical technical consideration with network IP cameras is time synchronization across cameras and true frame capture time reporting. Cameras are synchronized over *network time protocol* (NTP) to a primary and secondary stratum 1 GNSS-based time servers on the local network (in the network hub building) and frequently re-synchronize (roughly every 15 minutes). The camera firmware provides timestamps associated with video frames corresponding at about 10 microsecond accuracy relative to the camera clock time. Cameras capture up to 4K resolution video at 30 frames per second. Frame-to-frame timing is typically observed to be uniform (33.3 ms), but in some cases non-negligible time differences result from duplicated or skipped frames (an artifact of camera exposure requirements as implemented in camera firmware.) Although the camera clocks are are precisely synchronized and the exact frame capture times are different for all devices,

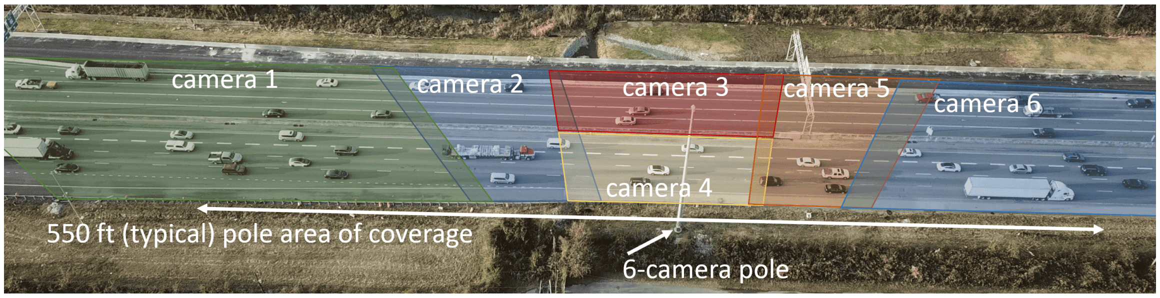


Figure 3.16: Example camera fields of view for a single 6-camera pole. Each portion of the roadway is covered by at least one camera, with overlaps long enough to allow objects to be tracked between cameras.

accurate time-stamping of each frame allows processing algorithms to compensate for the relative time offsets for each camera.

3.5.2 Network and Compute Hardware Architecture

All video data feeds are received from the TDOT network into a Vanderbilt data center for processing across a dedicated 40 gigabit fiber network connection. Centralized computing in a data center provides the computing hardware with dedicated, long-term support and infrastructure, in addition to future expansion possibilities. Two network switches support the cluster of servers: a data layer switch with two 25 gigabit connections to each server and a management layer switch for 1 gigabit user connectivity, control, and IPMI. A system control server directs the processing functions of the cluster across ten or more servers/nodes. It hosts a control interface where system managers dispatch processing jobs and propagates job configurations to each node. Nine processing nodes are dedicated to computer vision tracking and the initial trajectory construction. Each node contains eight graphics processing units (GPUs) that decode incoming video and perform object detection and tracking tasks. The nodes track vehicles across cameras, but each node operates independently with statically-assigned cameras. A vehicle traversing the entire instrument will generate a partial trajectory fragment on each of the (nine) processing nodes. Incoming video is buffered on its respective compute node and discarded after processing. Following initial trajectory generation, a post-processing server performs the complete trajectory assembly and reconciliation tasks. The cluster contains two data storage arrays responsible for storing the resulting trajectories – both initial trajectory fragments and post-processed complete trajectories – as well as log messages, monitoring data, algorithm training data, and instrument experiment data. Additionally, two servers within the cluster serve as a development and testing environment for new software versions and one server performs ancillary tasks such as large-scale visualization and traffic analysis.

3.5.3 Software Architecture

A prototype software architecture comprises of three main modules: video ingest, vehicle detection and tracking, and trajectory post-processing and reconstruction, managed by the system control server. Before a run session starts, related configuration files and metadata are registered and stored in database for record-keeping or re-processing.

3.5.3.1 Video ingestion and recording

The cameras produce a H.264 encoded video, currently at 1080p resolution and 30 frames per second to reduce the data size. The streams are split into 10 minute chunks and recorded into a Matroska (MKV) container. The timestamps, corresponding to the exact exposure moment of each frame (streamed separately in a custom field) which are incorporated into the PTS (Presentation timestamp) metadata during recording. This field is mandatory for video files, thus providing a standardized method for frame timing information, and enables interoperability with any conforming software. The video stream, with the current configuration and all 276 cameras, occupies $\sim$1 TB for each recorded hour at 1080p resolution.

3.5.3.2 Vehicle Detection and Tracking

Vehicle detection and tracking is performed using *Crop-based Tracking*, a joint detection and tracking method [91]. This method processes only cropped portions of each overall image, drastically reducing detection inference time relative to processing each frame fully. Implicit in the use of this method is an accurate object motion model; object priors from this motion model are used to produce cropping boxes for each object, and only crops are processed by the object detector on most frames. We use a Retinanet (ResNet-50 backbone) object detector [132] to detect car and truck classes as listed in Table III. Motorcycles are not currently detected but may be added in future work. For the motion model, a Kalman filter with linear dynamics is used (see Appendix G. Objects are assumed to travel with constant velocity along the primary direction of roadway travel, and are assumed to have zero velocity perpendicular to the primary roadway direction (note that this motion constraint is relaxed during data postprocessing and is only used during initial object tracking). The intersection-over-union metric is used to compute affinity between object positions and new detections [166]. IOU is computed based on vehicle footprints in space rather than bounding box coordinates within an image, which allows detections from multiple cameras with distinct fields of view to be incorporated provided accurate homography information is available for each camera (for more information

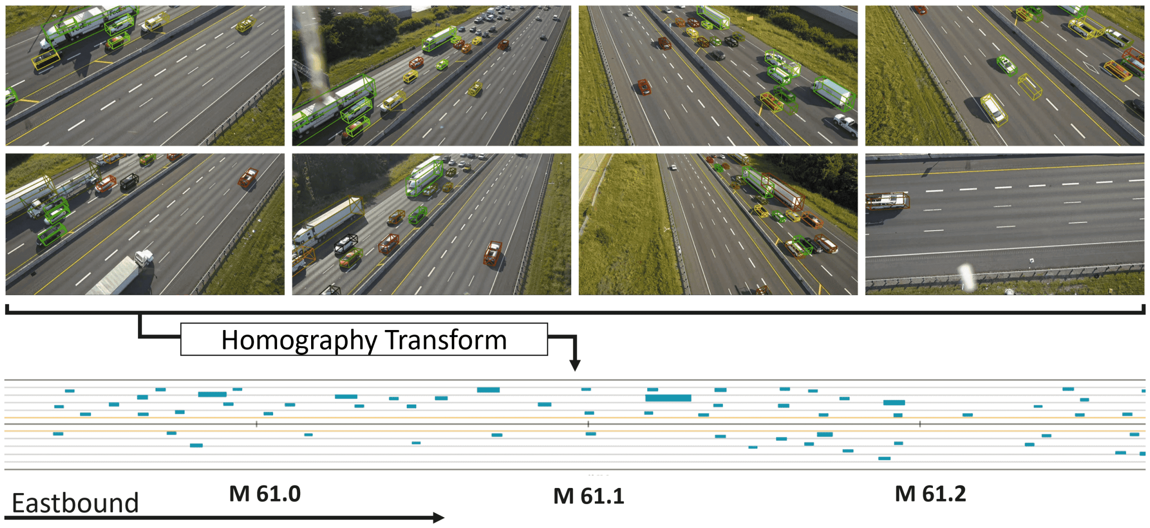


Figure 3.17: Vehicles are represented as 3D rectangular prism objects (various colors above) using object detection algorithms within each camera frame. The resulting detected objects are transformed into roadway coordinates shared among all cameras, and tracked in this unified coordinate system. Each blue rectangle in the projected 2D birds-eye view represents a vehicle position.

on camera homographies and data coordinate system, see Chapter 7. The multi-camera tracking problem is solved by detection fusion (as in [277, 279]) rather than trajectory fusion (as in [266]) to reduce redundant tracking of the same object in multiple fields of view. Figure 3.17 shows the result of object detection and tracking within image coordinates, and the corresponding roadway coordinate object positions obtained using image homography. The tracking algorithm is discussed in more detail in Chapter 5.

The complete set of 276 camera fields of view is subdivided across multiple processing nodes. On each node, all cameras are processed together (that is, roughly one frame from each camera is processed at a time, subject to some frame skips to keep cameras tightly time-synchronized). Processing nodes are not synchronized, so a single object traveling through the full instrument extents will be tracked as a separate vehicle with a unique ID on each processing node. This decouples the computation and allows the system to scale gracefully with a large number of cameras.

3.5.3.3 Trajectory Post-processing

Although raw trajectory data from dense deployment of cameras and CV algorithms can achieve complete spatial and temporal coverage of a roadway segment, such data contains inaccuracies from camera errors (dropped, doubled, and corrupted frames) network errors (data packet drops), object detection and tracking (fragmentations, ID swaps, false negatives and false positives [216]) often caused by object-object or infrastructure-object occlusions, timestamp quantization errors, homography assumption errors, and infeasible derivative quantities resulting from finite difference approximation over very short timescales. Treatments for specific sources of errors that rely on multiple iterations of rectification or require manual fine-tuning are not viable for longer term streaming datasets the I-24 MOTION is designed to produce. For small datasets, data cleaning and rectification with some manual involvement can address many common errors created in vehicular datasets [50].

I-24 MOTION uses an automatic data post-processing pipeline [311] which will be continuously improved to automate as much of the data cleaning steps as possible. Currently, it consists of a) an online data association algorithm to solve a min-cost flow problem, which consequently matches fragments that belong to the same object, and b) a trajectory reconciliation algorithm, which is formulated as a quadratic program. This algorithm reconstructs realistic vehicle dynamics from disturbed detection data with trajectory derivative smoothing and outlier correction while minimally perturbing the original vehicle detections. The resulting trajectories automatically satisfy the internal consistency (differentiation of trajectories with speeds and accelerations). Future post-processing development will consider conflict resolution along with trajectory smoothing to produce feasible inter-vehicular distances for accurate microscopic traffic studies, and may be able to leverage complementary efforts in trajectory prediction [312].

3.6 Trajectory Data

This section provides an overview of the data created by the I-24 MOTION system: its attributes, scale, conventions and coordinate system, known artifacts in the data, and a preliminary analysis of data accuracy.

3.6.1 Data Description

One single continuous recording session on the I-24 MOTION instrument processed through the software pipeline (from Chapter 3.5.3) results in a vehicle trajectory dataset. Each dataset produced by the system consists of a collection of individual vehicle trajectories. An individual vehicle trajectory consists of vehicle attributes as well as motion information (see Table 3.3). Trajectory positions record the 2D footprint of the back center of each car, and are re-sampled at a frequency of 25 Hz to allow exact timestamp-based indexing. Derivative quantities such as velocity, acceleration and steering angle can be directly computed with position information via, for example, finite difference. An example vehicle trajectory is included in Appendix C.

Attribute	Type	Unit	Description
_id	12-byte BSON	–	vehicle identifier unique across all I-24 MOTION data
vehicle class	int	–	*0: sedan*, *1: midsize*, *2: pickup*, *3: van*, *4: semi*, *5: truck*, *6: motorcycle*
first timestamp	float	s	minimum unix timestamp for this trajectory
last timestamp	float	s	maximum unix timestamp for this trajectory
timestamp	[float]	s	array of times at which vehicle positions are recorded
x position	[float]	ft	array of longitudinal positions on roadway corresponding to each timestamp
y position	[float]	ft	array of lateral positions on roadway corresponding to each timestamp
starting x	float	ft	longitudinal position on roadway at first timestamp
ending x	float	ft	longitudinal position on roadway at last timestamp
length	float	ft	vehicle length
width	float	ft	vehicle width
height	float	ft	vehicle height
direction	int	–	-1 if westbound, 1 if eastbound
configuration ID	int	–	identifier linking data to a unique metadata indicating trajectory generation algorithm settings

Table 3.3: Data attributes for a single vehicle trajectory. Square brackets indicate an array of values.

Accompanying this work, 10 days of trajectory data are released from weekday morning traffic. Each dataset spans typically 4 hours, from 6:00 AM to 10:00 AM, covering morning rush hour conditions. (Data from Friday, November 25th instead covers 11 hours.) A variety of traffic conditions are present throughout the various days of data, including at least three crash-induced bottlenecks, one debris-induced bottleneck, high-traffic conditions with travelling waves, and free-flow traffic conditions. Table 3.4 summarizes the data released with this work. Additional metrics, statistics, time-space diagrams, and useful information can be found with the data release, as this information will change as the data is updated in future versions. Time-space diagrams for the westbound portion of the roadway on each day of trajectory data are included in Appendix D. Details on the data release are included in Chapter 3.6.5. Weather data for the days of this data release can be obtained from the the *National Weather Service* (NWS) at https://www.weather.gov/wrh/Climate?wfo=ohx or *National Oceanic and Atmospheric Administration* (NOAA) at https://data.noaa.gov/onestop/.

3.6.2 Data Coordinate System

Data is provided natively in a curvilinear 2D roadway coordinate system, with the primary (x) axis aligned along the interstate roadway median and the secondary (y) axis defined locally perpendicular to the primary axis (see Appendix O). This means that x is roughly equivalent to station or mile marker along the roadway, while y gives lateral or lane-position data. A second-order spline defines the x-axis in global (state plane) coordinates. (Control points for the center-line in state plane coordinates are included in metadata). This allows for the direct conversion of roadway coordinates into state plane coordinates, with a trivial conversion

Date	Day	ID	Start time (AM)	Duration (hours)	Notes
Nov 21, 2022	Monday	637b023440527bf2daa5932f	6:00	4	crash, debris induced bottleneck
Nov 22, 2022	Tuesday	637c399add50d54aa5af0cf4	6:00	4	–
Nov 23, 2022	Wednesday	637d8ea678f0cb97981425dd	6:00	4	crash
Nov 24, 2022	Thursday	637f0d5f78f0cb97981425de	6:00	4	low traffic volume (holiday)
Nov 25, 2022	Friday	6380728cdd50d54aa5af0cf5	6:00	11	low traffic volume (holiday)
Nov 28, 2022	Monday	638450a3dd50d54aa5af0cf6	6:00	4	stopped vehicles induced slowdown
Nov 29, 2022	Tuesday	63858a2cfb3ff533c12df166	6:00	4	–
Nov 30, 2022	Wednesday	6386d89efb3ff533c12df167	6:00	4	–
Dec 1, 2022	Thursday	63882be478f0cb97981425df	6:00	4	merge induced slowdown
Dec 2, 2022	Friday	63898d48d430891009401330	6:00	4	crash

Table 3.4: Details of the released dataset. "ID" indicates the unique dataset identifier used to associate all data and metadata for this dataset. Additional summary and statistic information is included with the data release.

from state plane coordinates to GNSS WGS84 coordinates. Both coordinate directions are stored natively in feet. The positive x-direction is defined in the eastbound direction (direction of increasing post-mile as defined by the Interstate 24 mile markers), and x-coordinates are offset such that the x-coordinate for post-mile 60 corresponds exactly to $5280 \times 60 = 316800$ ft. (Other postmiles are approximately but not exactly located in this way (e.g. post-mile 61 $\approx 5280 \times 61 = 322080$ ft.) Adopting the left-hand rule convention, the y-coordinate is positive on the eastbound side of the roadway (vehicle is moving in increasing x-direction). Figure 3.18 illustrates the coordinate system.

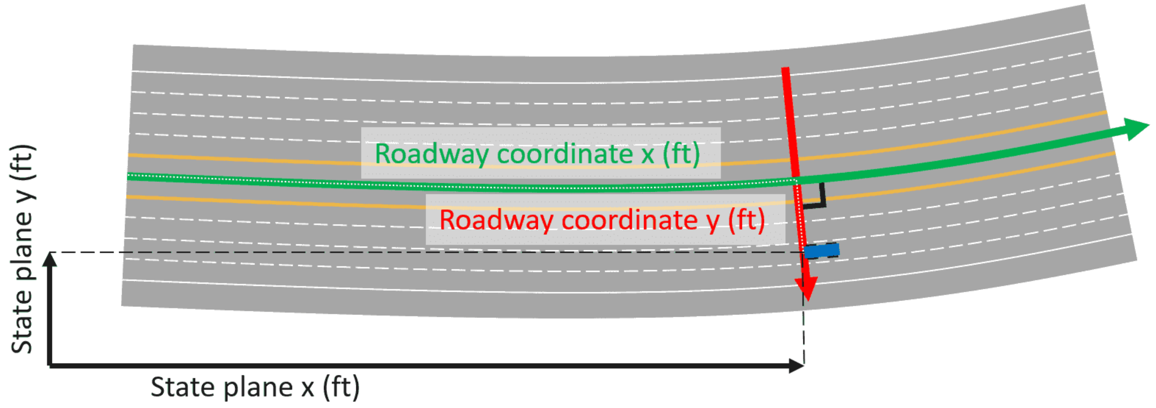


Figure 3.18: Spline-curvilinear x-axis (green) and locally perpendicular y-axis (red) for roadway coordinates. State plane coordinates are shown in black for comparison. Position of the vehicle can be expressed either in state plane coordinates (black dashes) or roadway coordinates (white dots).

The primary advantages of a curvilinear coordinate system are twofold: *i*.) The coordinate system aligns lateral (lane position) information along the y-axis, while accounting for the longitudinal curvature of the roadway and aligning the direction of travel with the x-axis. *ii*.) A perpendicular slice of the roadway has a uniform x-coordinate.

While definition of the y-axis as locally perpendicular to the x-axis does allow for the same point to have multiple (x, y) locations, for reasonable roadway curvatures these points occur suitably far from the roadway surface where the coordinate system is relevant. This coordinate system also slightly underestimates the distance travelled (and therefore the instantaneous speed) of vehicles on the exterior of a curve, relative to vehicles on the interior of a curve. the magnitude of this effect is no more than the ratio of roadway width to radius of curvature, which tends to be small (less than 5%) on typical roadways. Exact distances travelled and speeds can instead be calculated by converting positions into state plane coordinates followed by finite difference calculation. Precise roadway geometry including lane marking coordinates will be made publicly available in a future work.

3.6.3 Positional Accuracy

To assess the accuracy and suitability of I-24 MOTION trajectory data for micro-scale traffic analysis, output trajectory data is compared against an internal, manually labeled ground truth trajectory dataset, and onboard GNSS information from instrumented vehicles traveling on the roadway.

3.6.3.1 Manually Labeled Ground Truth

Manual labeling of vehicles as 3D rectangular prism bounding boxes within videos from a subset of 18 cameras was performed for two scenarios: a free-flow traffic scenario and a highly congested (one side of roadway) scenario. In total, over 600,000 individual vehicle positions were labeled manually. This data is described in more detail in Chapter 6. The resulting vehicle trajectories were compared against the trajectory data output by running the I-24 MOTION trajectory generation algorithms on the same video data. For comparison, object positions were matched to *ground truth* (GT) object positions as in [174] at each timestep. A minimum *intersection-over-union* (IOU) between the detected and ground truth vehicle position was required to consider the detected vehicle position a match for that ground truth object. Table 3.5 reports a number of multiple object tracking metrics for each scenario, as well as some metrics indicating the physical feasibility of the output trajectories. 97-98% of ground truth objects have at least one detected trajectory assigned to them (GT Match Rate) and for ground truth objects, on average 91-95% of the overall trajectory is covered by matching detected vehicle positions (Per GT Recall). Moreover, all vehicle accelerations produced by I-24 MOTION are physically feasible ($< 10 ft/s^2$), only 0-2% of vehicle observations have infeasible heading angles, and only 0-2% of vehicle trajectories overlap with another trajectory at some point.

Metric (1.0 best)	**Congested**	**Free-flow**	**Description**
MOTA	0.93	0.93	Aggregate object tracking metric
MOTP (IOU)	0.73	0.72	Average precision (IOU) of matched object positions
Precision	0.98	0.97	Proportion of detected object positions matched to a ground truth position
Recall	0.95	0.96	Proportion of ground truth object positions matched to a detected object position
GT Match Rate	0.97	0.98	Proportion of ground truth trajectories matched to at least one detected trajectory
Pred Match Rate	0.99	0.76	Proportion of detected trajectories matched to at least one ground truth trajectory
Per GT Recall	0.91	0.95	Average proportion of a ground truth trajectory with correctly matched detected object positions
Per Pred Precision	0.98	0.74	Average proportion of detected trajectory correctly matched to a ground truth object
Feas. Accel.	1.00	1.00	Proportion of finite difference accelerations that are feasible ($< 10 ft/s^2$)
Feas. Heading Angle	0.98	1.00	Proportion of finite difference heading angles that are feasible ($< 30°$)
Feas. Direction	0.99	1.00	Proportion of finite difference velocities with correct magnitude (no backwards movement)
Feas. Overlapping	0.98	1.00	Proportion of detected trajectories that never overlap with another trajectory

Table 3.5: Multiple object tracking and trajectory feasibility metrics for two ground truth scenarios (congested and free flow).

For matched vehicle positions, Figure 3.19 shows the relative error between the detected and ground truth vehicle position. 84% of detected vehicle positions fall within 3 feet of the ground truth position, and 36% fall within 1 foot of the corresponding ground truth. Table 3.6 reports the relative error between the detected and ground truth vehicle dimensions. All dimensions have a mean absolute error of less than 1.2 feet.

Quantity	**Mean Error (ft)**	**Standard Deviation(ft)**	**Mean Absolute Error (ft)**
Longitudinal (X) Position	0.2	2.6	1.7
Lateral (Y) Position	-0.3	0.6	0.6
Length	-0.6	2.5	1.2
Width	0.1	0.5	0.3
Height	0.5	0.8	0.7

Table 3.6: I-24 MOTION vehicle position and dimension errors relative to matched ground truth vehicles.

3.6.3.2 GNSS Data

Trajectory data was compared against onboard vehicle GNSS sensor data, a commonly used sensor modality for obtaining single vehicle trajectories. GNSS-equipped vehicles were driven in eastbound and westbound

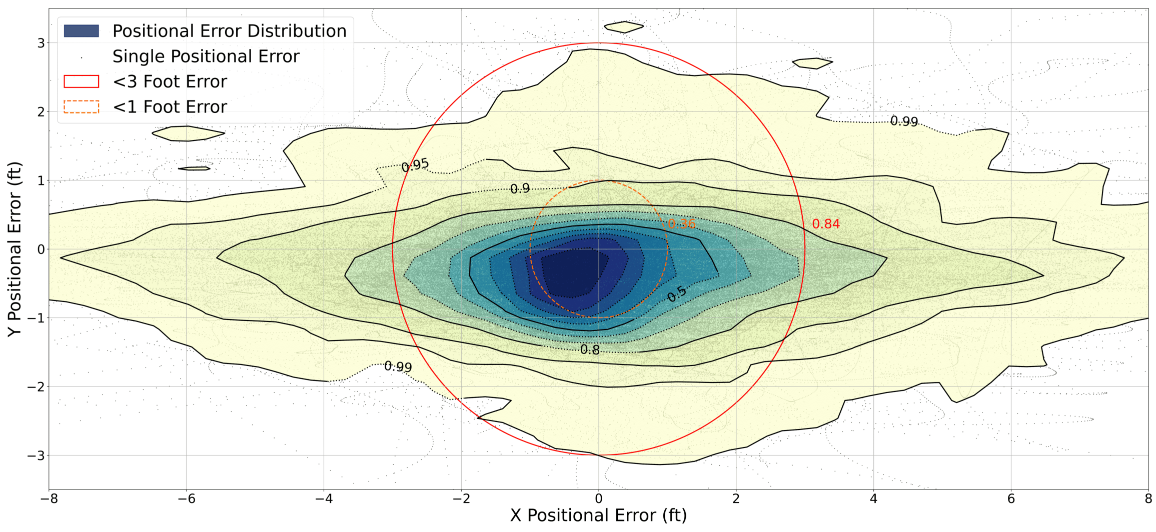


Figure 3.19: Positional error histogram for trajectory data relative to ground truth trajectories. Contours show the proportion of data is contained within, and are at intervals of 0.1 unless otherwise indicated. Single positional errors are shown as black dots. A red circle shows the proportion of data with less than 1-meter positional error (0.87) and an orange circle shows the proportion of data with less than 1-foot positional error (0.36).

lanes of traffic on the I-24 MOTION instrument [13]. Over 600 vehicle runs through the instrument extents were conducted. Regular (1 sec) positional data for each vehicle run was recorded. The reported *circular error probable* (CEP) for the sensor was 2.5 meters. Figure 3.20 shows a histogram of lateral positional data for each sensor modality (I-24 MOTION and GNSS data), aggregated for several longitudinal slices along the instrument. The I-24 MOTION data shows strong lateral peaks corresponding to vehicle presence in a specific lane of travel, whereas the GNSS lateral positional data does not show this characteristic. This is a strong indicator that I-24 MOTION yields strong lane-positional data, whereas this data is not necessarily available from an onboard GNSS sensor without heavy filtering. Due to this noisy lateral GNSS sensor data and the relatively high GNSS device error (2.5m CEP), we prefer the manually labeled vehicle trajectories over GNSS sensor data for validating the quality of I-24 MOTION trajectory accuracy.

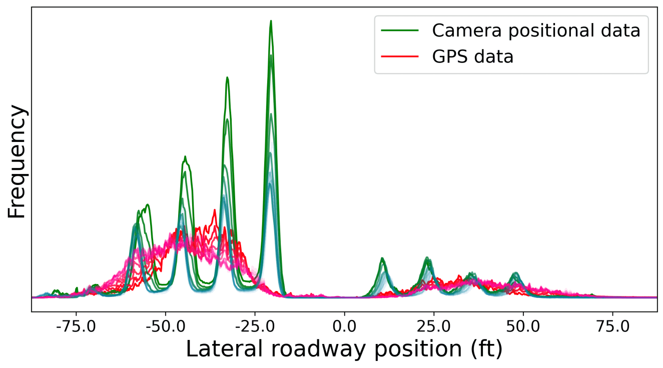


Figure 3.20: Lateral position histogram aggregated over several 1000-foot longitudinal slices, for I-24 MOTION camera trajectory data (blue-green) and onboard GNSS data (pink-red). Strong peaks I-24 MOTION camera positional data correspond to lanes of travel. Data produced during AM rush-hour (higher traffic volume on westbound, negative lateral position, side of roadway).

3.6.4 Data Artifacts

Relative to previous complete vehicle trajectory datasets, the data and instrument proposed in this work offer new challenges to perfect the data. Previous works were conducted in areas of sufficiently small spatio-temporal scale that physical occlusions could mostly be avoided (by overhead vantage point and careful roadway segment selection). Moreover, they were of sufficiently small temporal scale that errors remaining in the data after trajectory generation could be removed with manual efforts [50]). This approach is not scalable to the I-24 MOTION data, and some errors will always remain in the final data regardless of the algorithm employed. Enumerated here are a number of known errors in the initial data release that are artifacts of system hardware and software errors. We intend to partially or fully address each of these artifacts; moreover, open communication with I-24 MOTION data users will be maintained such that systematic errors in data creation can be addressed and data quality can be iteratively improved over time.

Figure 3.21 shows time-space data with each type of data artifact present. Known data artifacts include:

- **Missing Pole:** Data from a single pole is occasionally missing from one of two sources. Brief outages can occur due to network communication issues. or to physical hardware damage (a camera pole was hit by a car in the week prior to most of the data in this work being generated). This manifests as a horizontal band on the time-space diagram (a contiguous spatial range of data missing across all recording time). Such issues are rare because poles are protected by guardrails, but these issues cannot be eliminated entirely.

- **Overpass Occlusion:** Overpass occlusion results in lost tracked vehicles, which also manifests as a contiguous spatial range of data missing across all recording time. This artifact will be addressed with an intelligent data processing step that matches objects disappearing under bridges with objects reappearing on the other side.

- **Static Homography Errors:** Initially, homographies for each camera were statically defined. However, pole deflection due to temperature and sunlight cause subtle shifts in camera positions. This manifests in very narrow (a few feet wide) horizontal bands on the time-space diagram that contain missing or doubled trajectory positional data. This issue will be corrected by periodically accounting for subtle camera motion by re-defining homographies.

- **Packet Drops / Frame Corruptions:** Network bandwidth limitations (especially near night-time hours when low light conditions create noisier and therefore larger video data) result in occasional packet drops or frame corruptions, which manifest as a band of missing positional data for a contiguous region of space and time. This issue will be mostly addressed by IP camera stream profile optimization and network connectivity improvements.

- **Fragmentations:** Ideally, each vehicle passing through the instrument is represented by a single recorded trajectory. In practice though, vehicles are often represented by several trajectory fragments, which are often the product of the above artifacts or other tracking or post-processing failures. Fragmentations manifest as discontinuous chunks of trajectory corresponding to a single vehicle. Fragmentations will be iteratively decreased over time as the above artifacts and other tracking issues are removed.

3.6.5 Data Availability

I-24 MOTION is made available on the project website located at https://i24motion.org/data. Data will be associated with a DOI for permanent referencing, and new versions of data will be assigned new DOIs according to standard DOI issuing guidelines. A README file contains information relevant to downloading, formatting, and using the data. Each processed day of data (a JSON set of JSON-like trajectories) is made available for download, as well as additional metadata including: scene homography for the data, trajectory extraction algorithm settings, and in-depth descriptions of data attributes. Data is initially released "as is", recognizing over time the data will be reprocessed and improved as the instrument matures. New versions of this dataset will be updated as notable changes occur. Additional datasets (e.g., detailed lane markings) will be documented and released at the project website at https://i24motion.org/data) as they become available.

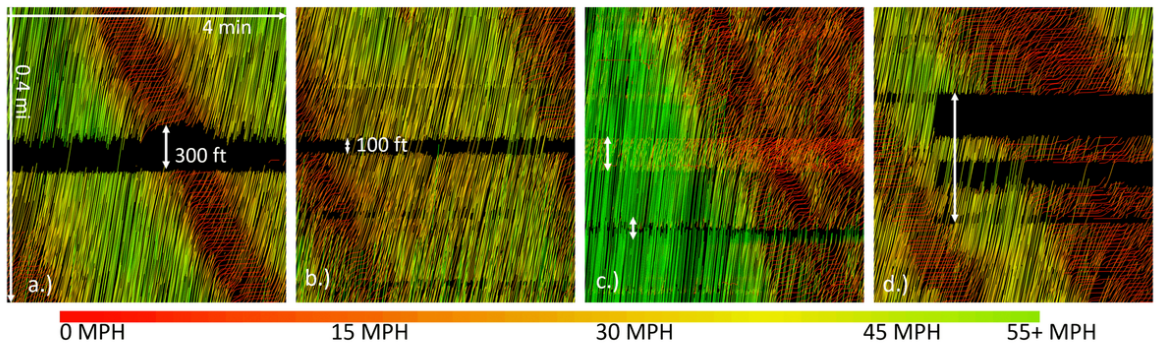


Figure 3.21: Example artifacts. For all figures, horizontal scale = 4 min. and vertical scale = 0.4 mi. a.) Missing pole causes a wide band of missing data. b.) Overpass causes a narrow band of missing data. In some cases post-processing can successfully stitch trajectories through this occlusion. c.) Homography error causes multiple trajectories corresponding to the same vehicle, or else results in a narrow band with no coverage. d.) Packet drops cause bands of missing trajectory data with a discrete start and end. Post-processing only partially fills in this data.

Video data is in general not persistently recorded or made available with trajectory data. This is because the raw video data potentially contains *personally identifiable information*. The instrument and data processing was designed to avoid collecting PII but it is difficult to guarantee no information was collected for all but very small subsets of data. We also note that the size of raw video files from the entire instrument is too large for easy distribution. For example the initial data release corresponds to approximately 47TB of video files. Depending on research community needs and IRB considerations, it is possible this may be reconsidered in the future. Moreover, two special-case datasets are released and detailed in chapters 6 and 7.

3.7 Preliminary Data Analysis

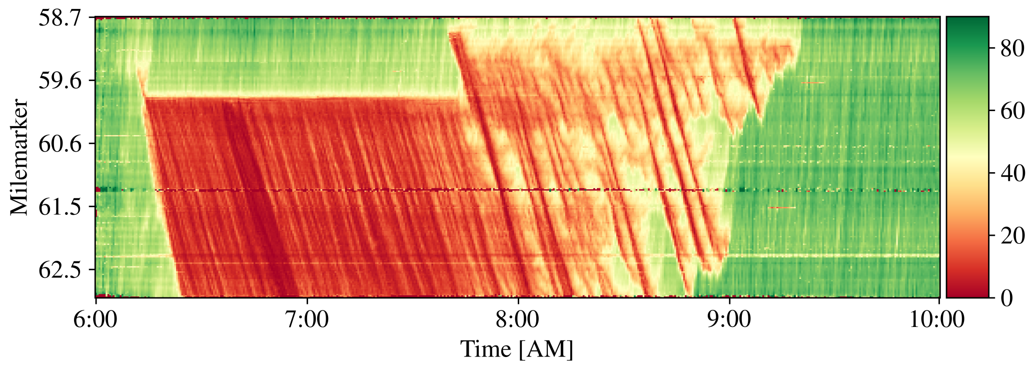
This section provides some initial analysis of the datasets that are publicly released. We generate the time-space diagrams of all of the published datasets, as well as illustrations of the type of analysis that can be conducted on the current data.

3.7.1 Traffic wave properties

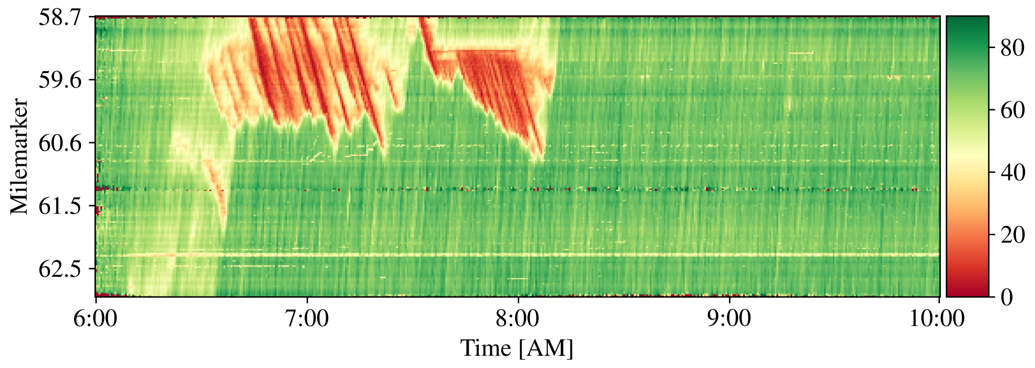
Traffic oscillations are characterized by regular acceleration/deceleration cycles in congested traffic, and is shown to have negative impact on the overall traffic efficiency and energy consumption [10, 25]. In this subsection we provide a few examples of macroscopic observations from a dataset captured by the I-24 MOTION system during the morning rush hours of two weekdays (Nov. 21 and Nov. 23, 2022) containing muiltiple events. The time space diagrams for these days are shown in Figure 3.22) including a variety of traffic patterns, such as free-flow, congested and stop-and-go traffic as well as bottlenecks caused by various incidents.

We select three signature events from these days (termed as Events A-C, see Table 3.7), which are incident-induced bottlenecks. Specifically, Event A is a severe rear-end crash on the HOV lane that was immediately followed by an onset of upstream queuing on lane 1 and lane 2. The congestion lasted for about 1.5 hrs before the crash was cleared. Event B is a slowdown on lane 3 caused by a large object falling out of a pickup truck. The roadway was cleared about 2.5 minutes later. Event C is a sideswipe crash due to a vehicle changing from lane 1 to lane 2 that caused a collision with another car travelling in lane 2. These events are summarized in Table 3.7.

Characteristics of the waves upstream of the selected events are calculated and also summarized in Table 3.7, including the wave propagation speed, period (time it takes to experience a complete slowdown and speedup cycle at a fixed location), and amplitude (or fluctuation range). Here the wave property calculations are based on visual inspections combined with various well-known techniques such as wavelet trans-



(a) Monday Nov. 21 2022



(b) Wednesday Nov. 23 2022

Figure 3.22: Velocity field in (mph) obtained from the westbound (decreasing milemarkers) trajectory data on (a) Nov. 21 and (b) Nov. 23, 2022. Each plot depicts traffic velocity evolution during the morning rush hours on the 4-mile of I-24 MOTION main corridor. The velocity field is aggregated into small bins from trajectory data according to Edie's definitions [2] with grid size of $\Delta t = 30$s and $\Delta x = 100$ft, respectively. The window sizes are selected to preserve fine-scale traffic wave properties. (Credit: Yanbing Wang)

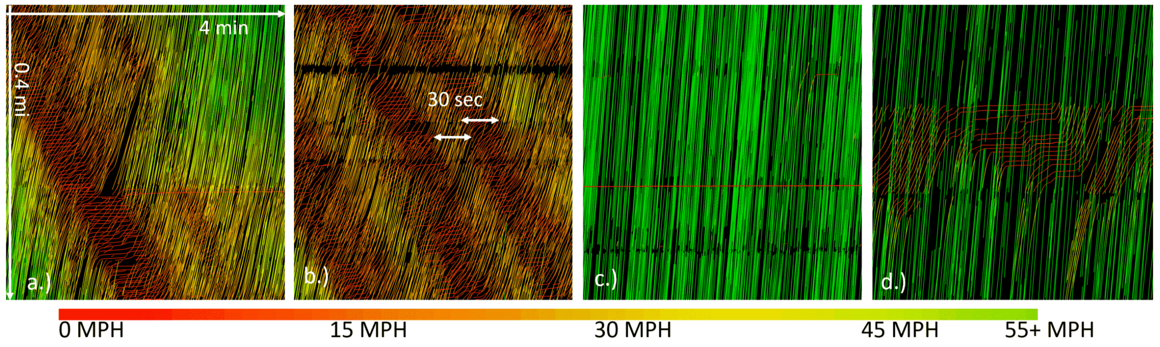


Figure 3.23: Examples of data phenomena difficult to observe in fixed-point or sparse GNSS floating vehicle sensing schemes. For all figures, horizontal scale = 4 min. and vertical scale = 0.4 mi. a.) Vehicle collision and resulting small-scale bottleneck. b.) Low-wavelength ($\approx$ 30 sec) traffic waves in high-density flow. c.) A stopped vehicle on side of roadway. d.) Off-ramp queuing during otherwise free-flow conditions.

form [313] and cross-correlation [314]. We direct interested readers to common references such as [33, 314] for details.

Figure 3.22 shows that perturbations in different times and locations all propagate upstream. Although the periodicity and magnitude of the waves vary, depending on factors such as the severity of the bottleneck, road geometry, and heterogeneity of driver-vehicle units [314], they generally travel against the direction of traffic at a constant characteristic speed of approximately 13 mph (see also [31, 315, 316]). We observe that oscillations with longer periods are often accompanied by larger amplitudes. For example, Event A has prominent waves with period 2.1 min and a speed range of 14.8 mph, Event B with period 5 min and a speed range of 34 mph, and Event C with period 1.8 min and a speed range of 10.8 mph, although the severity and the traffic conditions vary. The strong correlation between traffic wave period and amplitude is also discussed in [317].

Even in the present form, data from I-24 MOTION already suitable to study traffic waves and other macroscopic quantities. This allows I-24 MOTION data to be used for speed analysis directly without needing to extrapolate long distances between fixed sensors (data cleaning is, however still required). Moreover, the camera-based sensors yield useful insight into the initial causes of bottlenecks not visible in any other sensing modality (e.g., debris on the roadway). Figure 3.23 shows other example traffic phenomena not easily visible in traditional traffic sensing regimes.

3.7.2 Fundamental diagrams (FDs)

The empirical data from I-24 MOTION provides high resolution spatial-temporal evolution of traffic, which allows us to investigate more closely the changes of traffic properties on a finer scale. It also provides the possibility of computing fundamental diagrams at arbitrary locations around incidents.

Event Information						Upstream Wave Properties		
Index	Date	Duration	Nearest Milemarker	Description	Blocked Lanes	Propagation Speed (mph)	Period (min)	Fluctuation range (mph)
A	Nov 21	6:14-7:43AM	MM59.7	Severe rear-end accident	1,2 and left shoulder	12.6	2.1	0-14.8
B	Nov 21	7:40-7:44AM	MM58.8	Debris in lane	3	12.5	5.0	8.4-42.5
C	Nov 23	7:35-7:45AM	MM59.2	Sideswipe accident	1 & 2	13.1	1.8	8.7-19.5

Table 3.7: Approximate traffic wave properties in the upstream segment of selected events. The wave properties are obtained by a combination of wavelet transform and visual inspection (see Appendix F). Almost all waves appear to be “quasi-periodic” and non-stationary and therefore only the most prominent values are reported.

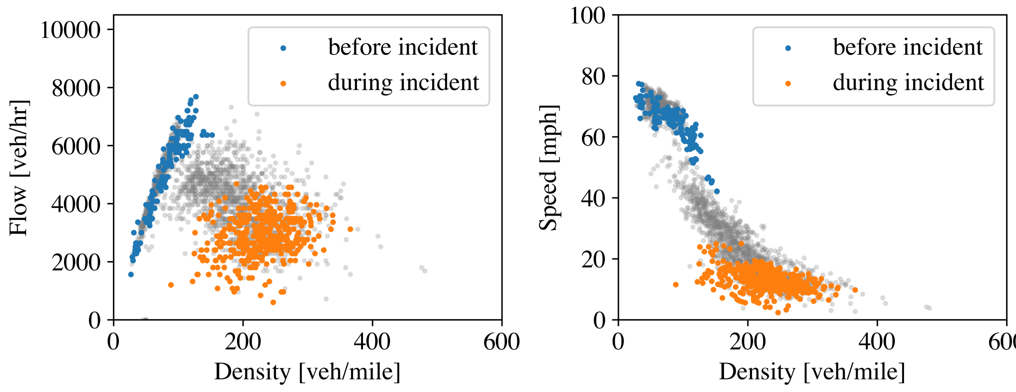


Figure 3.24: Flow, density and speed of west bound traffic at MM59.7 before and during event A on Monday Nov 21. (Credit: Yanbing Wang).

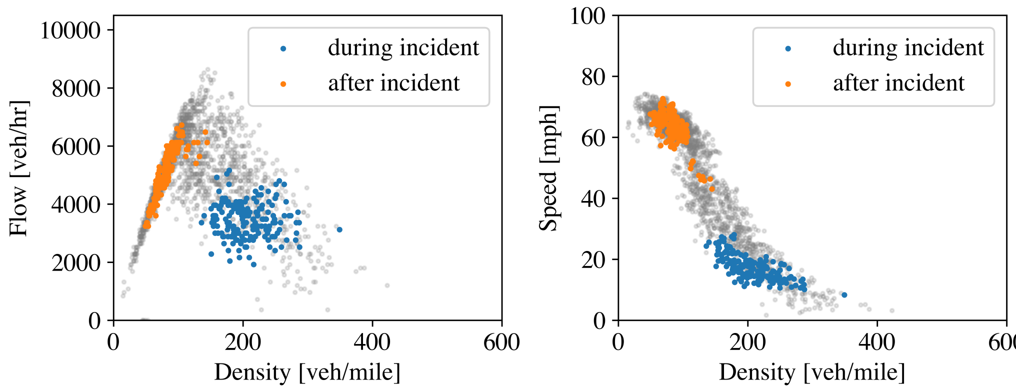


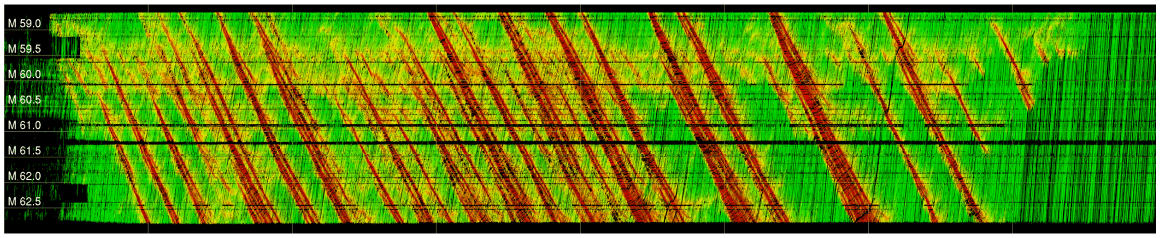
Figure 3.25: Flow, density and speed of west bound traffic at MM59.2 during and after event C on Wednesday Nov 23. (Credit: Yanbing Wang).

For example, Figure 3.24-3.25 show the flow, density and speed relationship at approximately 1000 ft upstream of event A and C, respectively, where there is 4 lanes of traffic. Specifically, the grey points in Figure 3.24 show all the traffic data at MM 59.7 during the 4-hr recording period from 6:00-10:00AM; the blue points correspond to the traffic data at the same location from 6:00AM to 6:15AM, immediately before the crash event A; the orange points show the most congested 15 minutes during the incident. Similarly, the blue points in Figure 3.25 represent the traffic data from 7:45AM to 8:00AM at MM59.2 during event C, the orange color corresponds to a 15-minute interval after the congestion is cleared, and the grey points are all the traffic data at MM59.2. The points are computed from the trajectory data using Edie's definitions. This illustrates a capability that is possible to explore precisely because the complete roadway is monitored, allowing us to analyze the data around each event location.

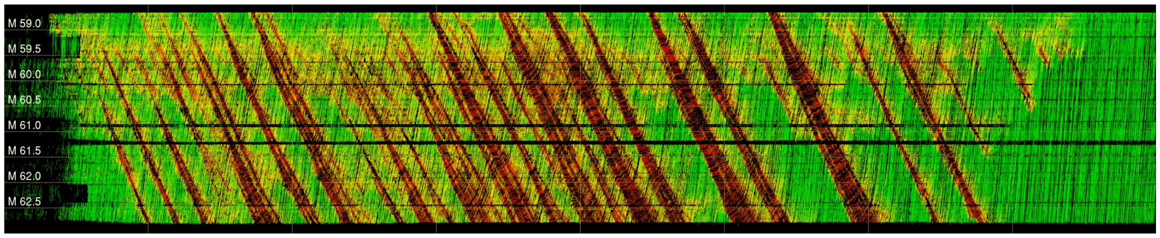
3.7.3 Lane-level wave analysis

The data from I-24 MOTION allows us to explore how waves propagate lane by lane. In Figure 3.26, we show the time space diagrams associated with the traffic conditions recorded on November 30, 2022. The data is shown by lane. Lane 1 is a high occupancy vehicle lane and is furthest from the freeway merges and diverges. Lanes increase in number from left to right where Lane 4 handles all vehicles merging into the freeway or exiting from the freeway. The images are colored on a red-green color-scale to better highlight the wave structure, with red associated with the slow moving traffic and green associated with fast traffic.

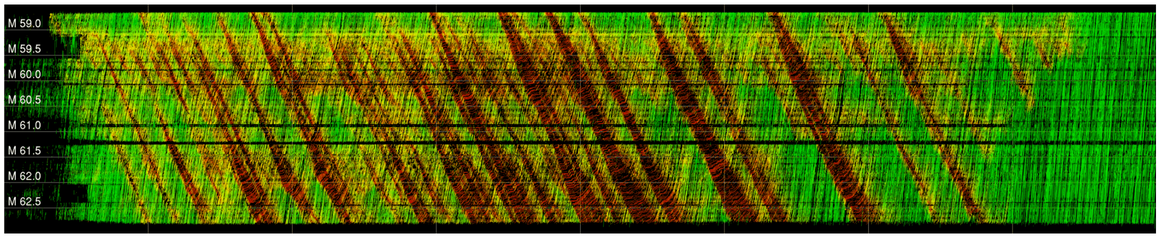
Comparing Lane 1 data (Fig. 3.26(a)) to Lane 4 data (Fig. 3.26(d)), it can be seen that the waves tend to be disrupted in Lane 4. In Lane 1, waves travel without disruption the full length of the roadway after they are formed. In contrast, waves in Lane 4 are disrupted and reform at multiple locations. As more data is collected, it will be interesting to determine if these patterns are repeated and if the mechanism to explain these patterns can be identified.



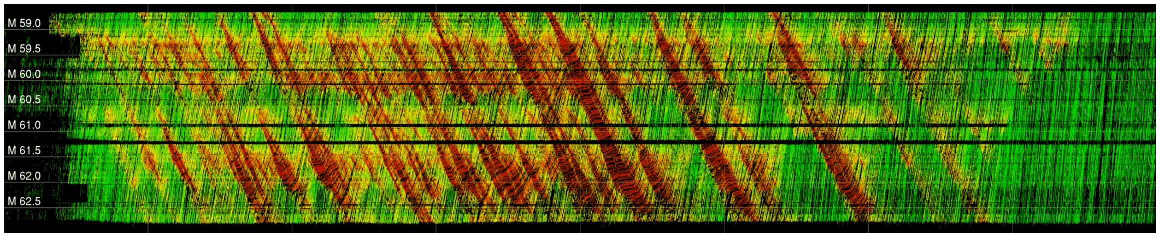
(a) Lane 1 (HOV Lane)



(b) Lane 2



(c) Lane 3



(d) Lane 4

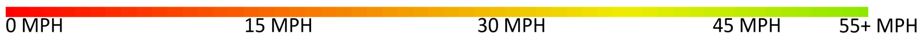


Figure 3.26: Wednesday, Nov 30 2022, 6:00-10:00 AM traffic waves by lane visible in time space diagrams. Figure (a) Lane 1 (High Occupancy Vehicle lane); (b) Lane 2; (c) Lane 3; (d) Lane 4 (closest to entrance and exit ramps). Traffic speeds are shaded green (fastest) to red (slowest). Lane positions are roughly approximated as constant lateral ranges, estimated by averaging the lateral coordinates of all lane markings for each lane. Comparing the waves in Lane 1 (a) to Lane 4 (d), the waves appear to travel the further and without interruption in Lane 1. (Credit: Gergely Zachar).

The current illustrations provided here are not comprehensive but are rather designed to show that the data in its current form can already be used to support different research questions. As the datasets continue to improve, it will allow further investigations that bridge microscopic and macroscopic scales. It may also allow labeling of vehicle trajectories under level 1 automated vehicle velocity control, for example using unsupervised methods [318]. Many of the best selling vehicles in the US have adaptive cruise control as a standard or optional feature now for several years, and consequently they are likely already in the datasets contained in this work. Labeling these vehicles could further aid understanding of the interactions between automated vehicles and human piloted ones.

3.8 Conclusion

This Chapter introduced the I-24 MOTION instrument, a system capable of producing large scale trajectory datasets to support new directions in traffic science and traffic flow theory research. We also provide our initial datasets that will be improved and maintained as the instrument software continues to mature. Next, Chapters 4 and 5 detail the unique algorithms developed for this demanding application.

4. Single-Camera Crop-based Tracking

4.1 Introduction

This chapter describes Crop-based Tracking, a novel method of detecting objects in small crops from overall video frames rather than processing each frame in its entirety. [1]

This work addresses the task of *multiple object tracking* (MOT) from raw video sequences in a traffic monitoring context. The goal of this task is to accurately localize and classify each object within a scene at each frame in the video, and to associate these bounding boxes across frames to provide matched identities for each unique object across time. In particular it considers fixed traffic cameras with overhead fields of view (as in the UA-DETRAC dataset [215]), which is important but distinct from the self-driving car context (e.g., the KITTI dataset [81]) in which the camera moves and is taken from a vehicle-centric field of view. The real-time performance of object detection and tracking is paramount for a number of tasks in traffic modeling (a shortage of vehicle trajectory data is a persistent problem [49, 66]) and to enable *intelligent transportation systems* (ITS) that dynamically respond to real-time demand to better accommodate traffic [319, 320]. Moreover, this task is strongly motivated by the needs of the I-24 MOTION system described in Chapter 3, which can only produce data on a recurring basis if the tracking algorithms implemented are able to process video data in realtime or near-realtime.

The vast majority of algorithms for multiple object tracking decompose the problem into two distinct tasks: First, the *object detection* task locates relevant objects within a frame. Second, the *object association* (commonly referred to simply as object tracking) task associates or matches objects in the current frame with the same objects in the previous frame such that each object is uniquely identified across the entire video sequence. A variety of recent methods [86–88, 90, 170, 207, 213, 222] have sought to leverage the generic tracking context to provide additional information for the object detection task, performing detection and tracking *jointly* rather than separately. These methods make use of the the relationship between objects in consecutive frames to boost object detection and tracking accuracy. Even so, most existing multiple object tracking methods are not well-posed to provide tracking data in real-time as they cannot on a single GPU process frames of modest size (e.g., 960×540 [1], 1392×512 [81], and 1920×1080 [82] in real-time .

In this work, a method is proposed to leverage the tracking context to further increase the speed of object detection and tracking. Especially in the traffic monitoring domain, objects have predictable motion, meaning that extremely strong priors for object locations within a video frame are available before that frame is processed. Generalized object tracking methods [5, 86–90, 166, 170, 181, 191, 206, 207, 211–214, 222, 321–323] cannot take advantage of object priors to reduce their object search space because they are intended to also track in contexts where camera and object motion is unpredictable.

4.1.1 Problem Formulation

The goal of object detection and tracking is to accurately localize and classify each object within a scene at each frame in the video, and to associate these bounding boxes across frames to provide matched identities for each unique object across time. The formulation and method of evaluation for this problem are detailed.

Formulation. The well-studied computer vision task of *multiple object tracking* (MOT) is formulated as follows. Let $\mathcal{O} = \{o_0, o_1, ..., o_m\}$ denote the set of all m objects of interest visible within a video with frames indexed by n in the range $[0, F]$. A single object annotation $o_i = [o_{i,0}, o_{i,1}, ..., o_{i,t}]$ consists of annotations for that object for each discrete time at which that object is visible within the frame. Each object-time annotation $o_{i,n} = [x_{i,n}, y_{i,n}, w_{i,n}, h_{i,n}, class_i]$ consists of a 2D rectangular bounding box parameterized by box center coordinate (x, y), box width w and height h, and an object class (e.g. car, bus, truck) constant for all annotations of object i. Object annotations need not be recorded for frames at which that object is outside of the frame. The goal of multiple-camera multiple object task is to predict $\mathcal{O}$.

Evaluation. Next, let $\mathcal{P} = \{p_0, p_1, ..., p_n\}$ be the set of predicted objects with the same formulation. Predicted outputs $\mathcal{P}$ are evaluated against true objects $\mathcal{O}$ on a per-frame basis. That is, for a frame n, the set of predicted objects for frame n are assigned to ground-truth objects for frame n (according to bounding box overlap),

[1]This chapter is adapted from [91].

then evaluated according to a number of metrics. These MOT metrics are described extensively in [216]. For simplicity, the de facto aggregate performance metric MOTA (PR-MOTA) is utilized as the primary indicator of tracking performance in this work.

4.1.2 Algorithm

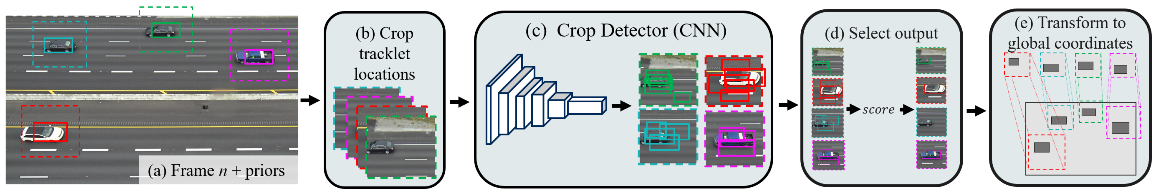


Figure 4.1: Overview of Tracking with Crop-based Detection (proposed). **(a)** Tracklet *a priori* locations (solid boxes) are used to **(b)**. crop (dashed boxes) around likely object locations. **(c)**. Detection is performed on crops and **(d)** the best detector output is selected for each crop.**(e)** The resulting bounding boxes (solid boxes) are then transformed back into the frame coordinate system, producing final object detections for each tracklet.

Figure 4.1 summarizes the proposed algorithmic approach. Consider an arbitrary *object detector* which takes an image as input and outputs a set of object detections for that image, and an arbitrary *object tracker* which assigns these detections to existing or new tracked objects. The proposed method, *Crop-based Tracking*, extends this detector-tracker pair by reformulating the task as a set of object detections performed on a set of images, each cropped (Figure 4.1 (b)) from the original frame (a) based on the location of tracked object *priors* with locations estimated utilizing previous frame information. (c) The object detector is used to detect possible objects in each crop (d) For each crop, the corresponding object prior is utilized to weight each output *before* removing any outputs from consideration as would be done in existing methods (with *non-maximal suppression* (NMS) or soft NMS [324]). The selected object bounding box for each crop is then (e) converted into their corresponding locations within the overall frame. Thus, on most frames, no association step is needed to process detections. Each step is explained in more detail for an arbitrary frame n next.

(a) Obtain inputs The method takes as input for frame n object priors computed with information from frames $0,...,n-1$, as well as frame n itself.

(b) Crop frame. Generate a square cropping box centered on each *a priori* object location in frame n. To ensure that the full object is contained within the crop, expand the crop to be larger than the a priori object estimate.

More precisely, let $P := \{1,\cdots,i,\cdots,p_{max}\}$ be the set of all (predicted) tracked objects, indexed by i. The *a priori* (denoted by tilde) bounding box for object i in global (frame) coordinates is defined by the center x-coordinate $\tilde{x}_i^g$, the center y-coordinate $\tilde{y}_i^g$, the width $\tilde{w}_i^g$ and height $\tilde{h}_i^g$. Let $\tilde{\text{box}}_i^g := [\tilde{x}_i^g, \tilde{y}_i^g, \tilde{w}_i^g, \tilde{h}_i^g]$. Similarly, we define the corresponding square crop for object i as $\text{crop}_i := [\tilde{x}_i^g, \tilde{y}_i^g, s_i]$, where s_i is the scale. The scale is computed as:

$$s_i = \max\{\tilde{w}_i^g, \tilde{h}_i^g\} \times \beta, \tag{4.1}$$

where β is a box expansion ratio (a parameter) used to ensure the full object is within the crop. Figure 4.2 shows a graphical representation of crop generation for a single object. By construction, crop_i is of size $(s_i \times s_i)$ pixels. Before detection, each crop re-scaled to a size $(C \times C)$ pixels, where C is a constant across all crops.

(c) Detect in Crops. All image crops corresponding to *a priori* object locations are processed by the detector, which produces bounding boxes that estimate the location of the object within each crop. Given crop_i, the detector returns l_{max} bounding boxes indexed by j in the local crop coordinates. Each output is an estimated location of object i within the crop, defined by the object center, box width, and box height. The j-th bounding box output of the detector corresponding to crop_i is written as $\text{box}_{i,j}^l := [x_{i,j}^l, y_{i,j}^l, w_{i,j}^l, h_{i,j}^l, \text{conf}_{i,j}]$, where $\text{conf}_{i,j} \in [0,1]$ is the confidence of the j-th detector output associated with crop_i.

(d) Select outputs. Each box is scored with a weighted combination of detection confidence and IOU overlap with the object prior (IOU+Conf), thus incorporating information from the prior before removing any

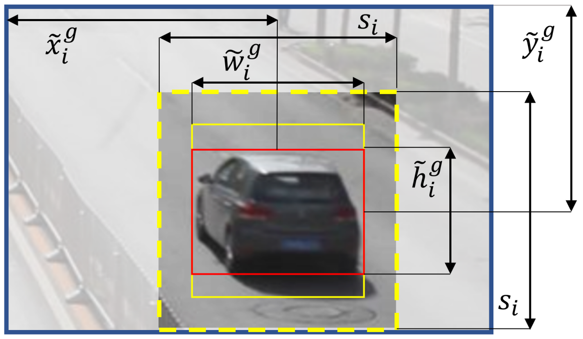


Figure 4.2: *a priori* object i location in global (frame) coordinates, $\widetilde{\text{box}}_i^g = [\tilde{x}_i^g, \tilde{y}_i^g, \tilde{w}_i^g, \tilde{h}_i^g]$ (red), is made square (yellow solid) and expanded by a factor of β to produce $\text{crop}_i := [\tilde{x}_i^g, \tilde{y}_i^g, s_i]$ (yellow dash) before being passed to the detector. Expansion helps to ensure the tracked object will be contained within the crop area.

candidate boxes. Section 4.1.3.2 describes the motivation for this choice. Each candidate bounding boxes is scored according to this IOU+Conf metric defined as:

$$\text{score}(\text{box}_{i,j}^l, \widetilde{\text{box}}_i^l) = W \times \text{conf}_{i,j} + (1 - W) \times \Phi(\text{box}_{i,j}^l, \widetilde{\text{box}}_i^l), \tag{4.2}$$

where Φ is the IOU similarity function between two boxes and W is a scalar used to balance the two terms. The bounding box with the highest score is selected as the detected box_i^l for object i.

The best detector output corresponding to crop_i is written as $\text{box}_i^l := [x_i^l, y_i^l, w_i^l, h_i^l]$, in coordinates local to the crop. Since the set of detection outputs for a crop are compared to the single *a priori* object i's location, output selection across all objects is $\mathbf{O}(o_{max} \times l_{max})$ in complexity, where o_{max} is the total number of tracked objects and l_{max} is the total number of detection outputs per crop. This operation is significantly less complex than a $\mathbf{O}(o_{max}^3)$ global min-cost matching problem in Step 3 of the base tracker [325]. Moreover it avoids object association errors that can occur in Step 3.
(e) Local to global transformation. The best detection box_i^l for each crop i is converted back into global coordinates, where it can be used to update the i-th tracklet. Any additional steps of the base tracking method are then performed (e.g. object book-keeping, prior prediction for next frame, etc.).
New Object Initialization The detector outputs for each crop a detection explicitly associated with an existing object, so inherently does not detect new objects. To initialize new objects, a second mode of operation called full-frame detection is added. Periodically, a full frame is processed by the detector rather than a set of crops. On these frames, the outputs are associated to existing objects or used to initialize new objects using the formulation of the base tracking method. All other frames are processed in the crop-detection mode.

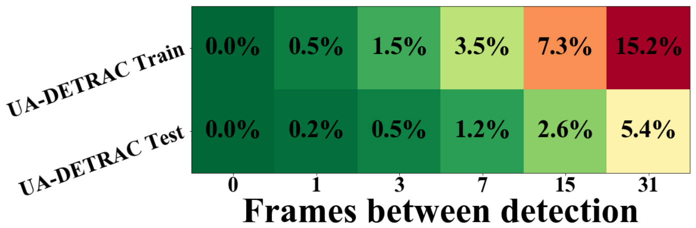


Figure 4.3: Expected increase false negative rate due to new undetected objects appearing between detection steps based on average object longevities for UA-DETRAC training and testing datasets. If every d frames is fully detected, on average a new object is missed in $\frac{d}{2}$ frames after it initially enters the field of view.

Because detection is not run on every full frame, there is a potential to increase the number of false negatives due to missed detections when objects first appear. However, Figure 4.3 shows that the expected increase in the false negative rate is small for real-world datasets. Moreover, we show in Section 4.1.3 that crop-based tracking improves overall tracking performance because of a dramatic reduction in false positives that appear when performing detection on every frame.

4.1.3 Experiments

This section briefly describes experiments conducted to assess the efficacy of Crop-based Tracking at boosting object detection and tracking speed. Section 4.1.3.1 describes experimental preliminaries. Section 4.1.3.2 describes experimental comparison of proposed versus existing detection thresholding and selection methods. Section 4.1.4 describes experimental analysis of Crop-based Tracking at different numbers of frames between full-frame detections. Section 4.1.4.1 details comparison of the proposed method against the state of the art. Section 4.1.4.2 describes the combination of Crop-based tracking with an existing tracking speedup method (frame-skipping).

4.1.3.1 Experimental Preliminaries

Base Algorithms. All subsequent experiments utilize an existing well-performing object detector (Retinanet) [132]. Two instances of the detector are trained, one for full frames and one for cropped frames, to account for differences in mean object sizes in the two modes of operation. A popular bipartite matching-based object tracker *Kalman-Filter enhanced Intersection-over-Union Tracker* (KIOU) [166] is used for the object tracker. KIOU estimates the position of each object with a Kalman filter (formulation described in Appendix G) and matches detections to existing objects by bipartite matching in terms of the overlapping area between the detection and object prior. The Crop-based Tracking extension of KIOU is referred to here as Crop-KIOU.
Dataset. All experiments are a performed on the UA-DETRAC dataset The UA-DETRAC Benchmark Suite, a traffic monitoring MOT dataset [215]. Training data is divided into training and validation partitions for internal experiments, and testing data is utilized only for final algorithm comparison against state of the art methods.

4.1.3.2 Evaluation of bounding box selection method

We test three strategies for selecting from amongst output bounding boxes for each frame. 1.) Perform *non-maximal suppression* (NMS) and subsequently select the remaining box with the highest IOU with the object prior location, evaluated at multiple threshold overlaps N_t. 2.) Same as 1, but use Soft-NMS instead of NMS, evaluated at several Gaussian factors σ [324]. 3.) The proposed IOU+Conf box-scoring described in Section 4.1.2 evaluated at several weighting parameters W. Figure 4.4 shows the resulting performance of each method over a variety of parameter values.

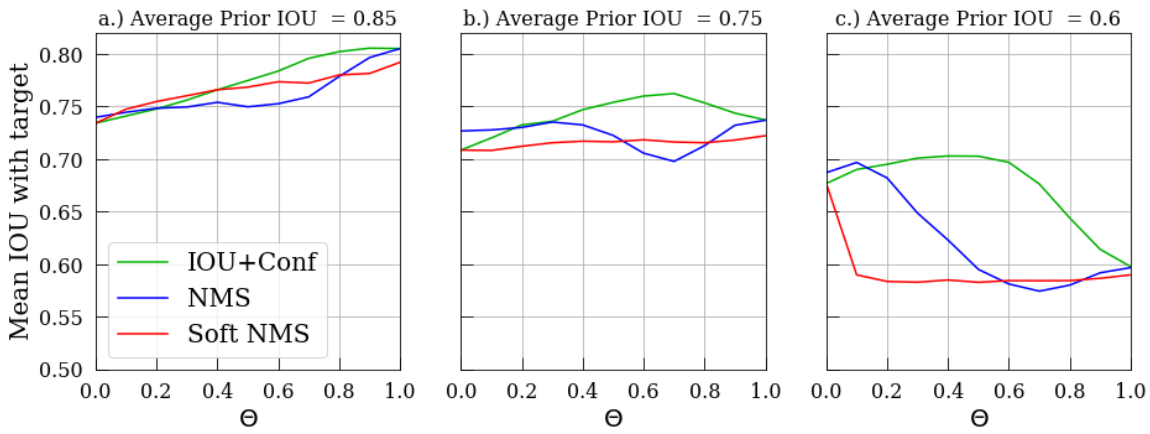


Figure 4.4: Mean IOU of ground truth and bounding box selected with each method, given object priors that overlap with ground truth by a.) 0.85, b.) 0.75, and c.) 0.60 on average. θ is a stand-in variable for the changeable parameter for each method (N_t for NMS, σ for soft NMS, and W for IOU+Conf).

The proposed IOU+conf approach outperforms both NMS and soft NMS at all tested levels of object prior accuracy. The best overall output IOU (0.81,0.76, and 0.70 respectively) is achieved using IOU+Conf for each object prior accuracy condition. Moreover, IOU+Conf performs reasonably well over a wide range of parameter W settings, indicating some robustness to suboptimal parameter assignment. IOU+Conf shows promise over NMS and Soft-NMS in constrained tracking contexts because the prior is incorporated earlier into the box selection process before any candidate boxes are deleted.

4.1.4 Crop-KIOU versus KIOU baseline

Crop-KIOU is evaluated for multiple object tracking on the UA-DETRAC training and validation partitions with varying numbers of frames between detection d. Tracking is performed at $d = 0, 1, 3, 7, 15$ and 31 frames. Note that $d = 0$ is the baseline (KIOU) because detection is performed on every full frame. Results are reported in Table 4.1.

d	Hz ↑	PR-MOTA ↑	PR-MOTP ↑	PR-MT ↑	PR-ML ↓	PR-IDS/id ↓	PR-FM/id ↓	PR-FP/obj ↓	PR-FN/obj ↓
0	22.7 / 22.9	66.4 / 55.8	77.5 / 70.7	**87.7% / 72.0%**	**3.9% / 8.6%**	0.27 / 0.45	**0.27 / 0.69**	0.192 / 0.220	**0.121 / 0.189**
1	26.4 / 26.7	**69.4 / 59.7**	**79.9 / 76.8**	73.4% / 53.7%	6.7% / 14.3%	0.51 / 0.64	0.90 / 1.35	0.093 / 0.090	0.190 / 0.275
3	28.5 / 29.1	67.2 / 58.0	78.1 / 75.7	70.0% / 50.7%	7.3% / 14.9%	0.43 / 0.53	1.13 / 1.39	0.095 / 0.094	0.208 / 0.287
7	31.3 / 31.9	63.9 / 56.6	76.6 / 75.4	63.3% / 44.4%	8.4% / 16.5%	0.35 / 0.39	0.95 / 1.15	0.098 / 0.092	0.236 / 0.306
15	32.5 / 33.1	60.7 / 53.6	76.0 / 74.9	52.2% / 34.9%	10.9% / 20.6%	0.20 / 0.23	0.67 / 0.85	0.096 / 0.089	0.272 / 0.338
31	**34.6 / 35.7**	55.6 / 47.6	75.7 / 74.2	36.8% / 25.2%	17.5% / 28.1%	**0.10 / 0.12**	0.49 / 0.65	**0.088 / 0.084**	0.337 / 0.397

Table 4.1: Tracking metrics (training/validation) for KIOU ($d = 0$) and Crop-KIOU on UA Detrac dataset. Best results for each metric are shown in bold. When applicable, metrics are normalized by number of unique objects (id), or number of unique object occurrences (obj)

As seen in Table 4.1, Crop-KIOU achieves increased accuracy (PR-MOTA) and increased frame-rate relative to the base tracker. When the number of frames between detection is small ($d = 1$ and $d = 3$), Crop-based Detection increases the overall accuracy (PR-MOTA) of KIOU by drastically reducing the number of false positives (PR-FP). Average tracking precision (PR-MOTP) is also increased, meaning more tracklets output by the tracker correspond to ground truth objects. Most importantly, Crop-based Detection also results in a speedup relative to the base tracker. At $d = 3$, a 25%/27% (train/validation) speedup relative to baseline is achieved in addition to an increase in accuracy. When $d = 7$ a 38%/39% speedup is achieved for a -2.5%/+0.8% change in PR-MOTA relative to the base tracker. This increase in speed does come at a slight penalty: False Negative rate (PR-FN), fragmentations (PR-FM), and identity switches (PR-IDS) all at first increase with increasing d. However, these types of errors are more desirable than false positives in a traffic monitoring context. Since object motion is fairly regular, missing object positions can often be imputed during post-processing, whereas false positive tracked objects that persist for several frames are more difficult to identify as anomalous.

4.1.4.1 UA-DETRAC testing results

Crop-KIOU (with $d = 7$) is compared to state of the art methods for the UA-DETRAC Benchmark test dataset as reported in [1]. Table 4.2 shows that Crop-KIOU outperforms all existing methods both on overall accuracy (PR-MOTA) and tracking precision (PR-MOTP). Additionally, Crop-KIOU performs best overall in terms of mostly tracked objects (50.1% and 41.1% PR-MT on beginner and advanced subsets, respectively.) Notably, Crop-KIOU has the lowest rate of false negatives (PR-FN) of any tracker on the advanced test dataset partition, and the second lowest PR-FN rate on the beginner partition. This means that, despite missing some new objects as they appear, Crop-KIOU tracks known objects accurately enough to produce very few false negatives, more than making up for missed new objects. Using Crop-based Detection to extend KIOU establishes a new state-of-the-art for this benchmark.

4.1.4.2 Comparison to frame skipping

Lastly, Crop-based Tracking is combined with *frame skipping*, the main existing method for speeding up object trackers. Frame skipping means performing object detection only every s frames and imputing tracked object locations between these frames. Combinations of crop-based tracking and frame-skipping with a variety of parameter settings for d and s are evaluated on the UA-DETRAC dataset in terms of framerate and MOTA, the aggregate MOT metric. Results are shown in Figure 4.5.

Crop-based Detection (green) extends the pareto-frontier of the accuracy-speed tradeoff for this traffic monitoring dataset. While frame-skipping (red) results in large speedups but exclusively reduces accuracy

Tracker	PR-MOTA ↑	PR-MOTP ↑	PR-MT ↑	PR-ML ↓	PR-IDs* ↓	PR-FM* ↓	PR-FP*↓	PR-FN*↓
GOG	23.9 / 11.7	47.4 / 34.4	20.5% / 10.8%	21.0% / 21.1%	0.0158 / 0.0124	0.0148 / 0.0119	0.119 / 0.123	0.70 / 0.70
IOUT	34.0 / 16.4	37.8 / 26.7	27.9% / 14.8%	20.4% / 18.2%	0.0109 / 0.0084	0.0115 / 0.0089	**0.031 / 0.061**	0.64 / 0.66
JTEGCTD	28.4 / 14.2	47.1 / 34.4	23.1% / 13.5%	18.3% / 18.7%	0.0013 / 0.0020	0.0050 / 0.0065	0.096 / 0.127	0.63 / 0.65
JDTIF	- / 28.0	- / 41.8	- / 34.2%	- / 20.9%	- / 0.0034	- / 0.0166	- / 0.270	- / 0.73
MFOMOT	34.6 / 14.8	46.6 / 35.6	30.2% / 11.9%	12.0% / 20.8%	0.0040 / 0.0042	0.0091 / 0.0098	0.073 / 0.103	0.52 / 0.73
KIOU	40.1 / 31.0	49.8 / 49.9	42.3% / 37.4%	**5.8% / 10.4%**	0.0021 / 0.0035	0.0024 / 0.0048	0.165 / 0.253	**0.25** / 0.46
V-IOU	37.9 / 29.0	41.7 / 35.8	38.1% / 30.1%	24.7% / 22.2%	**0.0004 / 0.0007**	**0.0008 / 0.0012**	0.073 / 0.069	0.66 / 0.70
DMC	- / 14.6	- / 34.1	- / 11.6%	- / 20.6%	- / 0.0044	- / 0.0062	- / 0.078	- / 0.68
GMMA	- / 12.3	- / 34.3	- / 10.8%	- / 21.0%	- / 0.0030	- / 0.0117	- / 0.124	- / 0.70
SCTrack-3L	25.9 / 12.1	47.2 / 35.0	15.0% / 7.7%	20.6% / 24.8%	0.0017 / 0.0018	0.0062 / 0.0046	**0.047 / 0.040**	0.74 / 0.79
Crop-KIOU (Ours)	**64.5 / 46.4**	**79.3 / 69.5**	**50.1% / 41.1%**	8.2% / 16.3%	0.0028 / 0.0051	0.0091 / 0.0186	0.061 / 0.113	0.26 / **0.44**

Table 4.2: Tracking metrics for UA-DETRAC Test Data (Beginner/Advanced) partitions. − indicates the result is not available. Results taken from [1].

* PR-IDs, PR-Frag, PR-FP and PR-FN are normalized by the total number of ground truth object detections.

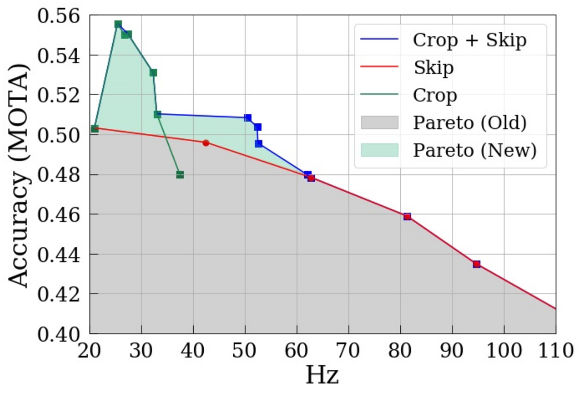


Figure 4.5: Fast Tracking method comparison. Each point represents tracking results at a single parameter setting. Crop-based Detection (green) and Crop-based Detection + Skipping (blue) extend the state of the art (Frame Skipping, red) in terms of the tradeoff between accuracy and speed (Hz).

relative to the base tracker, Crop-based Detection increases accuracy considerably (50.3 to 55.6 MOTA) while also increasing speed by 21% relative to the baseline tracker, or increases speed by 57% without decreasing accuracy. Furthermore, the combination of Crop-based Detection and frame-skipping (blue) results in even larger speedups (21.0 to 52.4 fps or 149%) without a decrease in overall accuracy relative to the KIOU baseline. This above real-time performance allows for multiple traffic cameras to be processed simultaneously on the same device in real-time.

4.2 Contribution: Vehicle Turning-Movement Counting with Crop-based Tracking without Detection

This section describes Vehicle Turning-Movement Counting with Crop-based Tracking, which proposes a novel extension of Crop-based Tracking that initializes new objects without full-frame detections and extends the method to the task of vehicle turning movement counting. [2]

4.2.1 Overview

Vehicle turning-movement counting is an essential tool for transportation planning. Accurate vehicle counts are necessary to determine roadway utilization, identify areas of congestion, and optimally allocate funding to maximally increase transportation quality of service within a constrained budget. Historically, these vehicle counts were performed manually using handheld electronic devices. The nexus of three key trends in the

[2]This section is adapted from [92].

past 10 years make vehicle turning-movement counting a problem of renewed interest, as seen by this task's presence in the 2020 and 2021 AI City Challenges [326]. First, the development of accurate CNN-based image processing methods [59] allow for video data to be reliably used to provide information in transportation contexts. Second, state and federal transportation organizations have become increasingly interested in *intelligent transportation systems* (ITS) that both utilize and incorporate traffic data to allocate resources and even make transportation decisions in real-time. [327]. Lastly, *edge computing* devices continue to become cheaper, more computationally powerful, and more ubiquitous such that they now provide a feasible tool by which to study traffic in many cities provided lightweight and accurate algorithms for counting can be developed [328, 329].

Existing algorithms rely on slow, full frame object detection [245, 248, 251–260]. This work proposes to leverage Crop-based Tracking to avoid slow detection steps during tracking [91]. The main contribution of this work is to utilize object source regions within a camera field of view and process these regions with the localizer at every frame such that detection is never performed on a full frame, yet new object tracklets can still be initialized at every frame. To the best of our knowledge, this proposed approach is the first traffic counting algorithm to explicitly avoid performing object detection on whole frames.

4.2.2 Problem Formulation

For a video sequence with frames indexed by n, the goal of the *multiple turning movement counting task* is to predict a set of vehicle turning movements $T = \{t_1, t_2, ..., t_m\}$. Each turning movement $t_i = [movement, n]$ is a specific turning movement (e.g. eastbound -¿ southbound left turn) selected from a discrete set of possible turning movements for a particular camera field of view, at a particular frame n.

4.2.3 Algorithm

The proposed method, Crop-Count, utilizes Crop-based Tracking method Crop-KIOU described in Section 4.1 as a base detection and tracking method, with 3 novel extensions:

i.) Source Region Crops for New Object Initialization Since cameras for traffic monitoring or on edge-equipped devices are relatively static, new vehicles appear in a few, well-known regions within each camera field of view. These source regions are manually labeled per field of view call the regions where new objects appear *source regions*. Source regions are manually identified once for each camera field of view. Figure 4.6 shows example source regions for a few camera fields of view. In addition to cropped regions based on existing object tracklets, each source region is also cropped to localize potential new vehicles. All crops are resized to square images of a standard size c_s pixels. The purpose of these source region crops is to allow the initialization of new objects without full-frame detection.

In each crop, regions containing visual information likely to mislead the crop-detector and reduce tracking accuracy are blacked out. These can include regions that are always misleading (e.g. parking lots and street-parked vehicles) and regions that are only misleading when initializing new objects (e.g. traffic on the opposing side of a highway). Regions of the former type are blacked out in all image crops, whereas regions of the latter type are blacked out only in crops corresponding to source regions such that existing objects can still be tracked through these regions. Figure 4.6 shows examples of each type of ignored region.

ii). New Object Initialization After detection, outputs for object-prior crops are parsed to select the best bounding box as in Crop-based Tracking [91]. Detector outputs corresponding to source region crops are parsed differently. Instead, the outputs for each source are parsed using the following logic:

1. All output bounding boxes with confidence lower than σ_{min} (a tuned parameter) or with a predicted class not in $\{car, truck\}$ are removed.
2. Non-maximal suppression is performed on all remaining bounding boxes.
3. All remaining bounding boxes with confidence lower than σ_{new} (a tuned parameter) are removed.
4. Any item that overlaps with an existing object by more than ϕ_{new} (a tuned parameter) in terms of intersection-over-union metric is removed.
5. All remaining bounding boxes are used to initialize new object tracklets, and the source region from which each object was initialized is recorded. The new object's speed is initialized in the Kalman filter as the estimated average speed of objects originating from the same source.

iii). Movement Counting Just as objects tend to enter a frame at a few source regions, objects also exit the frame in a few, well-known *sink regions* within each camera field of view. These sink regions are manually

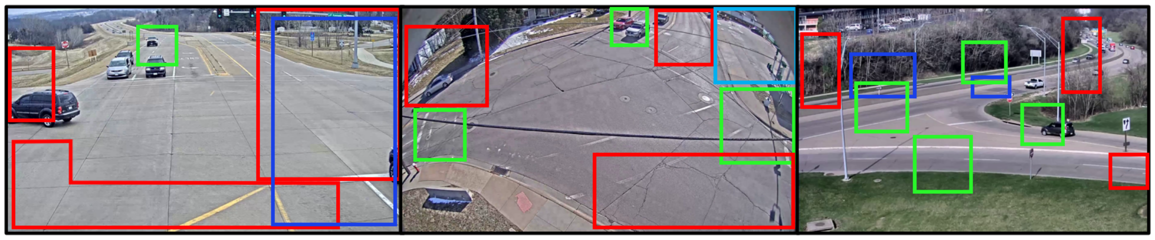


Figure 4.6: Examples of vehicle source regions (green), sink regions (red), regions that are blacked out in source crops only (dark blue) and regions that are blacked out in all crops (light blue) for several camera fields of view. Source regions are cropped and searched for new objects at each frame (Steps 2-3). Detected vehicles are tracked until they have travelled from a source region to a sink region, and the unique source-sink combination defines the vehicle's unique turning movement (Step 6).

labeled once for each camera field of view. After Step 5, each object is compared to each sink region. If the center of that object's bounding box falls within a sink region, that object is no longer tracked. Each unique source-sink combination identifies a vehicle turning movement of interest, so this corresponding vehicle movement is output by the algorithm. Example sink regions are shown for several camera fields of view in Figure 4.6.

Lastly, logical limits are imposed on the frequency with which specific vehicle movements can occur. The minimum number of frames between movements f_{move} is set per vehicle movement, per camera view, and only source-sink combinations corresponding to valid vehicle movements are recorded. This helps to avoid double-counting vehicles in the event that multiple object tracklets correspond to a single real vehicle or that a tracklet from one source mistakenly begins tracking a vehicle from another source.

4.2.4 Experiments

This section briefly describes experiments conducted to assess the efficacy of the propose Crop-Count algorithm for vehicle turning movement counting. at boosting object detection and tracking speed. Section 4.2.4.1 describes experimental preliminaries. Section 4.2.4.2 describes algorithm results on a turning movement counting challenge, and Section 4.1.4.1 explores the speedup of Crop-Count relative to a tracking-by-detection counting algorithm based on KIOU rather than Crop-KIOU.

4.2.4.1 Experimental Preliminaries

Dataset. The proposed method is evaluated on the 2021 AI City Challenge turning movement counting challenge, which requires multi-class, multi-movement vehicle counting on video sequences at intersections and along roadways. Thirty-one sequences from 20 distinct camera views are included, comprising about 9 hours of total video data all of which has resolution of at least 1280×960. Each camera field of view contains several vehicle movements of interest.
Evaluation. To motivate the design of algorithms that can be evaluated in real-time on edge compute devices, the computational efficiency of vehicle counting algorithms are taken into account in addition to counting accuracy for this challenge. Algorithms are scored both on accuracy, and on processing speed (efficiency).

4.2.4.2 AI City Challenge Results

The proposed algorithm places 7th out of 17 algorithms and the proposed work processes video at an average of 72.6 frames per second on a single GPU and 2 CPU cores. Figure 4.7 shows the path of each object that was counted as a valid vehicle movement for several sequences, with object paths of each vehicle movement colored uniquely. There are very few anomalous paths indicating that few objects experience identity switches and generally tracking is quite accurate, even on sequences with many distinct turning movements.

4.2.4.3 Speed Comparison to Tracking by Detection

Lastly, to benchmark the impact of using crop-based tracking rather than tracking-by-detection (TBD) for the object detection and tracking portions of our counting method, a detect-track-count algorithm based on KIOU

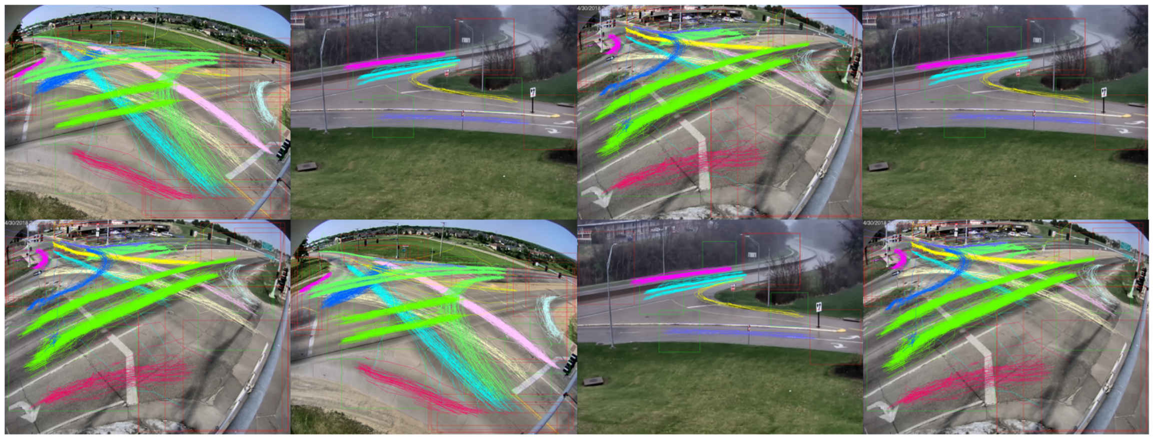


Figure 4.7: Vehicle paths for counted vehicle movements. Each movement is shown in a unique color per sequence. Faint green and red boxes denote source and sink regions, respectively.

object tracking [166, 322] is implemented. The speed of each method is measured when a measurement step is performed at every frame. The same network structure is used for the crop detector for the crop-based method and the full-frame detector in TBD.

The proposed algorithm is 52% faster than the detect-track-count (TBD) approach overall (20 fps vs 13.2 fps average), is faster than TBD on 29 of 31 available test sequences, and achieves at least a 100% speedup on 19 of 31 sequences. The speedup of the proposed method is somewhat correlated to the number of crops (the sum of the number of tracked objects and the number of source regions for a camera field of view), as each cropped region requires additional computation to localize vehicles within it. Sequences with fewer than 19 crops per frame on average exclusively experience an increase in speed.

Per-scene results and competition results from the 2021 AI City Challenge Turning Movement Count Track are given in Appendix H.

4.3 Conclusion

This section proposed a fast method for single-camera multiple object tracking and demonstrated its utility on a traffic monitoring dataset. In Chapter 5, a variety of techniques are proposed to extend this method to be suitable for multi-camera multiple object tracking.

5. Crop-based Multi-Camera 3D Tracking

This chapter describes Multi-camera Crop-based Tracking, which proposes a novel method for tracking objects across multiple cameras by limiting camera queries per object to increase tracking speed relative to Crop-based Tracking (described in Chapter 4) and state-of-the-art baselines. A novel loss function for 3D object detection, polygon intersection-over-union loss, is also introduced as an extension to this work.

5.1 Overview

Tracking objects precisely in a space viewed by multiple cameras introduces new challenges not present in the single-camera MOT task. First, a 3D representation of each object that uniquely captures its position is required (2D bounding boxes are insufficient as the position of the object in 3D space is ambiguous, depending on the viewpoint of the object). A 3D rectangular prism bounding box is widely adopted for vehicle representation [81, 124, 285]. While positionally precise, 3D bounding boxes are more difficult to predict because of the inherent ambiguity in mapping from a 2D space (image plane) to a 3D space. Existing approaches either predict bounding box corner coordinates natively in 2D image space [115, 148, 152], or predict objects on the ground plane in 3D space [286, 330, 331], which allows for the intelligent incorporation of physical constraints such as expected object sizes, but requires an explicit or implicitly learned homography transform and sacrifices generalizable object detection in other viewpoints.

The second challenge introduced by multiple-camera tracking is the requirement of a method to match views of the same tracked object across multiple cameras. Existing approaches solve this either by i.) unifying all input frames before performing object detection [261–264], unifying detections across all camera views before tracking objects [276, 277, 279, 280], or else by performing object tracking independently in each camera view and clustering trajectories from all camera views afterwards [118, 267–273]. The former set are online methods, while the latter are not.

In addition to these challenges, the multi-camera detection and tracking task introduces new contextual information that can aid in the tracking task. First, real-world physical information can be utilized to constrain predicted object tracklets to reasonable vehicle dimensions, speeds and accelerations [332]. Second, multiple views of the same object yield redundant or corroborating information on each vehicle's position. Existing works utilize this information by combining multiple measurements into a single refined estimate of an object's position [277]. Third, object persistence across overlapping camera views provides information about when and where new objects can appear and disappear within the overall shared space. Existing approaches utilize this information only very weakly to perform object hand-offs between camera views [275].

In this work, a new approach is proposed that tackles the multi-camera problems addressed above and utilizes each source of new contextual information to track objects more accurately and efficiently. This approach trains a 3D object detector in 2D image space to preserve generalizability across many camera views, and integrates information from multiple cameras by only utilizing information from a single camera, per object. This method is based off of the Crop-based Tracking proposed in Chapter 4 but utilizes multiple-view redundancy to further reduce computation.

5.1.1 Problem Formulation

Formulation. The multi-camera, multiple object tracking problem is similar in formulation to the MOT formulation described in Chapter 4. Let $\mathcal{O} = \{o_0, o_1, ..., o_m\}$ denote the set of all m objects of interest across all c camera views indexed by k, viewed at discrete times indexed by t in the range $[0, T]$. A single object annotation $o_i = [o_{i,0}, o_{i,1}, ..., o_{i,T}]$ consists of annotations for that object for each discrete time at which that object is visible within the field of view of any camera. Each object-time annotation $o_{i,t} = [x, y, \theta, l, w, h, class]$ is parameterized by a 3D-rectangular prism bounding box, located on the shared ground plane with center coordinates (x, y) (in feet), orientation (heading angle) θ, vehicle dimensional length, width, and height $[l, w, h]$ (in feet), and vehicle class $class$. The goal of multiple-camera multiple object task is to predict $\mathcal{O}$.

5.2 Approach: Multi-camera Crop-based Tracking

The proposed approach is an extension of the algorithm described in Chapter 4. A novel camera query selection step and a coordinate conversion step, are added to the original algorithm and are described in more detail next, yielding the overall formulation:

1. **Load Frames and Object Priors** - the set of frames at a fixed time t for all camera views is taken as input.
2. **Select query camera per object** - an object may be visible in multiple cameras; assign a single camera to each object to query for a measurement.
3. **Crop selected regions from overall frames** - each region corresponds to an object prior visible within the selected camera's field of view.
4. **Detect objects in crops** - crops from all camera frames can be processed simultaneous (i.e. as a batch).
5. **Convert objects from image space to shared 3D space** - utilizing scene homography.
6. **Select best detection for each crop** - based on Conf+IOU metric from Chapter 4.
7. **Bookkeeping** - Update filter measurement update, add and remove objects, etc.

5.2.1 Load Frames and Object Priors

Frame selection and prior location estimation is mostly straightforward, except for two details that make it not at all straightforward, namely: i.) frames from each camera are recorded "out of phase" (see Appendix J) meaning that for a given time t we can only hope to achieve a frame synchronization of $t \pm \frac{1}{2f}$, where f is the time between consecutive frames from a single camera. A nominal framerate is selected and at each processing time-step, the target time t is advanced exactly according to the nominal framerate. Then, for each camera frames are skipped until a frame with a timestamp less than $\frac{1}{2f}$ behind t is reached.

This causes ii.) for a nominal time t, the time Δ_o between the last known position of each object o and each frame to be distinct. To use a filter-based motion model such as a Kalman filter for object prior state predictions, we violate the assumption of constant time between measurements. Thus, the filter must be modified to account for the propagation in state uncertainty over varying time durations. Consider two cases: i.) *Perfectly time-correlated noise*, where state covariance scales exactly linearly with time, and ii.) *Perfectly uncorrelated "white" noise*, where uncertainty scales with the square of Δ_o. This result is derived in Appendix G.3. Case ii applies for an ideal Kalman filter, so we use this assumption in the state covariance update step. (Full model dynamics and state formulations are given in Appendix G.2.)

5.2.2 Selection of Query Camera

Rather than cropping each object from every camera frame in which that object is visible, this work proposes to crop only one region from one camera frame per object. In other words, we select a single camera to "query" for a measurement for each object. Let $Q_{i,k}$ denote a query score for tracked object i, camera k pair. The query score is computed as:

$$Q_{i,k} = F_{i,k} * T_k * \frac{1}{D_{i,k}} * P_k \tag{5.1}$$

where $F_{i,k}$ is 0 if object i is not within the field of view of camera k, and is 1 otherwise; $D_{i,k}$ is the real-world distance between the base of the camera pole for camera k and object i, T_k is 1 if the current frame from camera k has a valid timestamp (indicating successful frame decoding) and is 0 otherwise; and P_k is a priority score for camera k, set to 1000 for interior cameras ($C03andC04$) 100 for intermediate cameras ($C02$ and $C05$) and 1 for exterior cameras ($C01$ and $C06$) meant to prioritize cameras with top-down views of an object over cameras that view an object from a long field-of view (with greater occlusion and longitudinal position uncertainty).

For each object i, the query score is computed for all cameras and the camera with the highest overall query score is selected. Succinctly, this equation is summarized as: *select the camera with a successfully decoded frame that is closest to object i and has an un-occluded view, prioritizing an un-occluded view over distance from object.*

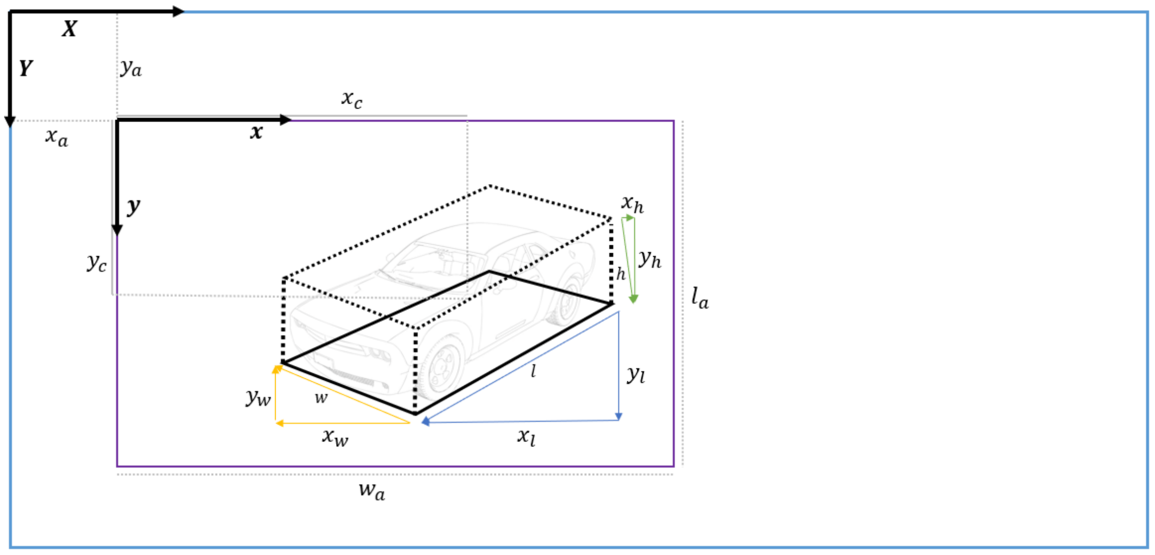


Figure 5.1: Bounding box *rectangular prism* parameterization. Anchor box (purple) reference coordinate system denoted as (**x**,**y**) and global frame (blue) reference coordinate system denoted as (**X**,**Y**). Angle of h is exaggerated to show sub-components x_h and y_h.

5.2.3 Detect objects in Crops

This step is similar to the analogous step from Chapter 4, with the exception that the object detector must be parameterized to produce 3D rectangular prism bounding boxes rather than 2D rectangular bounding boxes. Object detections are produced by a Retinanet object detector with Resnet-50 FPN backbone [132]. We extend the first convolutional layer of this network to allow two input frames (the current and a rolling average frame). This network has two output heads: a classification head and a regression head, each of which outputs a set of outputs per anchor box. The shape of the classification head output C is $[n,c]$ where n is the number of anchor boxes in the network and c is the number of classes. For an anchor box i, the final output object class and object confidence are taken as:

$$\text{class}_i = \arg\max_c(C[i,:]) \tag{5.2}$$

$$\text{confidence}_i = \max(C[i,:]) \tag{5.3}$$

The shape of the regression head output R is $[n,8]$. Let A_i parameterize a single anchor box i according to:

$$A[i,:] = [x_a, y_a, w_a, h_a] \tag{5.4}$$

denoting the x-coordinate,y-coordinate,width and height of the anchor box, respectively. The regression output corresponding to that anchor box $R[i,:]$ parameterizes a rectangular prism (ignoring the effects of perspective foreshortening) according to:

$$R[i,:] = [x_c, y_c, x_l, y_l, x_w, y_w, x_h, y_h] \tag{5.5}$$

where (x_c,y_c) indicates the center of the rectangular prism relative to the anchor box top left corner, x_l and y_l indicate the x-pixel and y-pixel components of the rectangular prism's length, x_w and y_w indicate the x-pixel and y-pixel components of the rectangular prism's width, and x_h and y_h indicate the x-pixel and y-pixel components of the rectangular prism's height. All components (e.g. x_l,y_l) are relative to the anchor box dimensions (in this case w_a and h_a). By convention, each dimension is directional and is measured relative to the rear bottom left corner. Figure 5.1 provides a visual overview of each component.

The (**X**,**Y**) pixel coordinates for back bottom left coordinate (x_{rbl}, y_{rbl} within the overall image can then be written as:

$$
\begin{aligned}
x_{bbl} &= x_a + (x_c - x_l/2 - x_w/2 + x_h/2) * w_a \\
y_{bbl} &= y_a + (y_c - y_l/2 - y_w/2 + y_h/2) * h_a
\end{aligned}
\tag{5.6}
$$

The 7 other corner coordinates can be similarly written. The resulting rectangular prism can then be converted into state plane and roadway coordinates via the methods discussed in Appendix O. Readers familiar with 3D detection in the autonomous vehicle context will likely question the use of 3D bounding boxes produced explicitly in state space, as in e.g. [143, 144]. In these contexts, a single camera is statically mapped to the world coordinate plane (i.e. the same pixel within an image always corresponds to the same point on the ground plane in world space). This is not the case in our application, in which a single CNN is desired to produce object detections for frames from all 234 cameras. Thus, a *viewpoint agnostic* anchor box / bounding box formulation such as the one used here is required. We also consider the direct regression of 3D bounding box corner coordinates as in [152] but empirically this results in poor performance.

5.2.3.1 Object Detector Training

We train the above object detector on the I24-3D dataset [95], described in more detail in Chapter 6. We use the following loss formulation:

$$
Loss = Loss_{2D} + \beta \times Loss_{3D} + Loss_{cls} + \gamma Loss_{vp} \tag{5.7}
$$

where:

- $Loss_{2D}$ is intersection-over-union loss [333] for the minimum enclosing 2D bounding boxes for each of the predicted and target bounding box
- $Loss_{3D}$ is MSE loss computed between the target and predicted bounding box corners
- $Loss_{cls}$ is classification focal loss [132]
- β is a weighting coefficient, here set to 2.
- γ is a weighting coefficient, here set to 1/3

and $Loss_{vp}$ is a term designed to enforce the length, width, and height components of the predicted rectangular prism to align closely to the corresponding vanishing points by penalizing the angle between these vectors and the vanishing point directions (relative to the center of the predicted bounding box). **Succinctly, $Loss_{vp}$ is 0 when each bounding box is perfectly axis-aligned relative to the vanishing points, and is 1 when the axes of the bounding box are perfectly orthogonal to their respective vanishing points.** In this way, the training enforces the CNN to utilize visual cues from the image and align bounding boxes along these visual axes.

$$
\begin{aligned}
Loss_{vp} =& 1/2 - \alpha_l((y_l(y_{vpl} - y_c)) + x_l(x_{vpl} - x_c))/(2(x_{vpl} - x_c)(y_{vpl} - y_c)y_l x_l) + \\
& 1/2 - \alpha_w((y_w(y_{vpw} - y_c)) + x_w(x_{vpw} - x_c))/(2(x_{vpw} - x_c)(y_{vpw} - y_c)y_w x_w) + \\
& 1/2 - \alpha_h((y_h(y_{vph} - y_c)) + x_h(x_{vph} - x_c))/(2(x_{vph} - x_c)(y_{vph} - y_c)y_h x_h)
\end{aligned}
\tag{5.8}
$$

where (x_{vpl},y_{vpl}) is the vanishing point, in pixel coordinates, aligned with the roadway direction of travel, (x_{vpw},y_{vpw}) is the vanishing point, in pixel coordinates, perpendicular with the roadway direction of travel, and (x_{vph},y_{vph}) is the vanishing point, in pixel coordinates for vertically aligned lines within the frame, α_l is -1 if the rear of the bounding box is closer to (x_{vpl},y_{vpl}) than the front of the bounding box and 1 otherwise, α_w is -1 if the left side of bounding box is closer to (x_{vpw},y_{vpw}) than the right side of the bounding box and 1 otherwise, and α_h is -1 if the bottom of the bounding box is closer to (x_{vph},y_{vph}) than the front of the bounding box and 1 otherwise.

Convert objects from image space to shared 3D space. The object detector outputs objects parameterized by bounding rectangular prism corner coordinates in image space. Camera homography information is utilized to convert object points on the ground plane into shared 3D space (lossless), and the height of the object in 3D space is selected to minimize the point reprojection error into image space. Then, the resulting corner coordinates in 3D space are (with some slight rectification error) converted into the simplified state formulation, rigidly constraining each object into a rectangular prism state representation as defined in Equation 5.9. The state of an object at frame n, $\mathbf{x_n}$, is expressed as:

$$\mathbf{x}_n = [x_n, y_n, l_n, w_n, h_n, v_n]^T, \tag{5.9}$$

where (x_n, y_n) is the back bottom rear center coordinate of the vehicle in roadway coordinate space, and (l_n, w_n, h_n) are the dimensions of the vehicle (length, width, and height in feet, respectively), and v_n is the object's velocity in the x-direction (in ft/sec). This state formulation is a relatively straightforward modification of that described for SORT [5] adapted to 3D space. Note that the heading angle of the vehicle θ is assumed to be zero to maintain a linear and observable dynamical system. The full 3D filter formulation is shown in Appendix G.

5.2.4 Select Best Detection for Each Object and Bookkeeping

The resulting objects have the same formulation as the Kalman filter formulation for tracked objects, so they can be used to i.) select the best detection for each crop or initialize new objects as in [91] and Chapter 4, and then to perform filter measurement updates. The remainder of the algorithm is identical to single-camera formulation in Chapter 4.

5.2.5 Experiments

The resulting multi-camera 3D object tracker is benchmarked on the I-24 3D dataset. This dataset and benchmarking results are described in Chapter 6.

5.3 Polygon Intersection-over-Union Loss

This section introduces a novel loss function for training viewpoint agnostic 3D object detectors such as the one described in the previous section. [1]

Autonomous driving is a primary domain that propels research in 3D object detection. Precise detection and localization of vehicles and pedestrians within a driving scenario are paramount to autonomous vehicles functioning safely and effectively. To enable this end, densely annotated ego-vehicle driving datasets produced with carefully calibrated and heavily instrumented test vehicles such as KITTI [160], NuScenes [285], Waymo OpenDrive [124] have enabled a large body of work on 3D object detection and tracking tasks. As a result, extremely accurate detection 3D vehicle, cyclist, and pedestrian models have been proposed that leverage the full suite of available sensors, including LIDAR, stereo images, and depth images [334] [335] [336] [337] [338] [339]. State-of-the-art 3D detection methods on KITTI frequently score above 90% AP_{70} (average precision) [160].

The dense sensor set provided in these datasets comes at a price. Methods proposed utilizing these works are not generalizable to other vehicles with different or less capable sensing. Recognizing this shortcoming, *monocular 3D object detection* methods have been proposed to predict object positions using a single camera and no additional sensors [143, 144, 150]. Posing the 3D object detection problem in this manner introduces the challenge of recovering depth information from an image, which is inherently depth-ambiguous. Monocular methods take a step in the direction of generality; only a single camera is required for detection; yet these methods incorporate information from the 3D scene explicitly (e.g., into the model anchor box generation architecture) or implicitly (by training to regress object positions directly in 3D space). Thus, a model trained for one camera in one vehicle can't be easily applied to another vehicle and camera, and training data is only available for a very small subset of instrumented vehicles.

To detect 3D objects from a single, arbitrary camera, such as a cell-phone camera or dashcam from an arbitrary vehicle, a more general method is required. Recently, a subset of monocular 3D detection methods have attempted to detect 3D bounding boxes for vehicles without utilizing scene information in the trained model. Instead, these models predict positions natively in 2D image space and incorporate scene *homography* only after training and after inference [151–153, 155]. By posing 3D detection in this way, a trained model

[1]This section is adapted from [94].

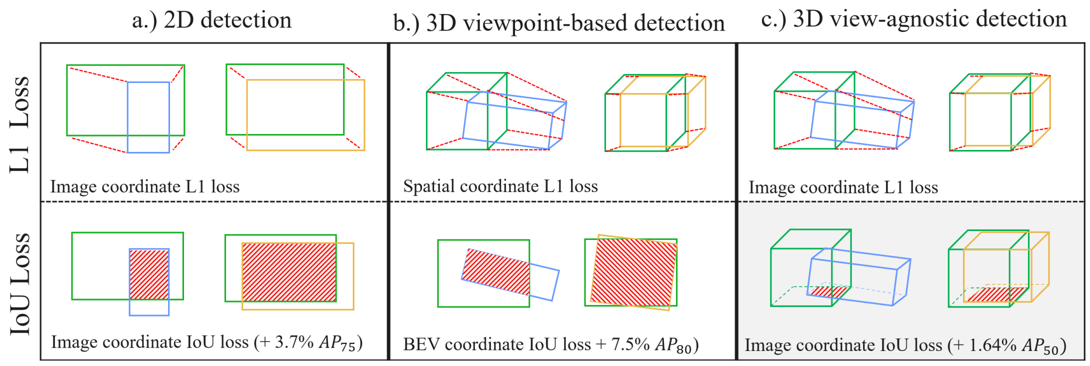

a.) 2D detection
b.) 3D viewpoint-based detection
c.) 3D view-agnostic detection
L1 Loss
IoU Loss
Image coordinate L1 loss
Spatial coordinate L1 loss
Image coordinate L1 loss
Image coordinate IoU loss (+ 3.7% AP_{75})
BEV coordinate IoU loss + 7.5% AP_{80})
Image coordinate IoU loss (+ 1.64% AP_{50})

Figure 5.2: Example of IoU-based losses and relative improvements versus L1-loss in a.) image coordinates, where bounding boxes are axis-aligned, [3] b.) 3D viewpoint-based detection, in which the scene homography can be used to precisely compute object rectangular footprints in a *bird's-eye view* [4] and c.) 3D viewpoint-agnostic detection (this work), using 3D box projections into 2D space.

may be able to generalize to an unseen camera view simply by changing the post-inference scene homography. In other words, these methods are *viewpoint-agnostic*. The performance of these models, while steadily improving, still trails other monocular 3D detection methods and 3D detection methods more generally.

This work seeks to leverage a key trend in object detection works: across a variety of domains and detection problem formulations, *intersection-over-union* (IoU) based methods have been shown to outperform L1 and L2 norm-based methods for loss calculation during training. Intersection over Union (IoU) is commonly used for measuring the similarity between (generally) two rectangular, axis-aligned bounding boxes. Many previous works on 2D object detection tasks demonstrate that IoU can be used as a regression loss for axis-aligned 3D bounding boxes. In [3] [157] and [158] it is shown that incorporating IoU loss into 2D object detection models can improve their performance. In a similar vein, [4] and [159] show that IoU for two rotated rectangles can be used as a loss function to improve the performance of 3D object detection models. Figure 5.2 summarizes. Unfortunately, these methods are not directly applicable to viewpoint-agnostic monocular methods because the projection of a 3D bounding box into an image does not result in rectangular planes; rather, the six surrounding planes of vehicles occupy arbitrary quadrilaterals in pixel-space. Thus, most existing methods use L1 loss to regress the eight corner points of the 3D box on 2D image planes.

The core contribution of this work is to present a new and efficient way of calculating the IoU between two convex polygons which we refer to as *polygon IoU* and implement it as a loss function (PIoU loss). We show both in simulation and in 3 state-of-the-art viewpoint-agnostic 3D detection models that the loss function converges faster than L1 loss. We implement a batched version of the IoU function between two polygons to enable fast training of models with the method. We utilize models trained with PIoU loss on the KITTI 3D detection benchmark and show that, in most cases, the new loss formulation increases model accuracy, particularly for higher requisite IoU thresholds.

The rest of this section is laid out as follows: Section 5.4 describes the PIoU method in more detail. Section 5.5 describes experiments comparing L1 loss and PIoU loss with simulated polygons, Section 5.6 details experiments on the KITTI benchmark and describes implementation details for incorporating PIoU loss into 3 detection models. Section 5.7 describes the results.

5.4 Polygon IoU Loss

The *Polygon IoU* (PIoU) method proposed calculates the intersection-over-union metric for any two convex polygons in 2D coordinates. The inputs are two sets $\mathcal{A}$ and $\mathcal{B}$ consisting of the (x, y) corner coordinates of each polygon, and the algorithm output falls in the range $[0, 1]$. This output can then be utilized as a loss function $Loss = 1 - PIoU$.

5.4.1 Overview

Polygon IoU loss calculation consists of:

1. Order the points of $\mathcal{A}$ and $\mathcal{B}$ clockwise.
2. Compute $\mathcal{C}$, the set of all points of intersection of any two edges of the polygons.
3. Find $\mathcal{A}_B$, the set of all points in $\mathcal{A}$ that lie in the interior of $\mathcal{B}$, and vice versa for $\mathcal{B}_A$.
4. Compute the area of the convex polygon defined by the overlapping set of points $\mathcal{I} = \mathcal{A}_B \cup \mathcal{B}_A \cup \mathcal{C}$
5. Compute the areas of $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{I}$.
6. Compute PIoU according to:

$$\frac{Area_{\mathcal{I}}}{Area_{\mathcal{A}} + Area_{\mathcal{B}} - Area_{\mathcal{I}}} \tag{5.10}$$

We describe each step in more detail below.

5.4.2 Clockwise

A set of points is ordered in a clockwise manner by computing the geometric center of the polygon. Then, angles are calculated between an arbitrary first point (defined to be at 0°), and each other point, relative to the geometric center. Points are then sorted in order of decreasing angle relative to the center. Note that the clockwise ordering of $\mathcal{A}$ and $\mathcal{B}$ is necessary for subsequent computational steps which assume a clockwise, geometrically adjacent ordering of points.

5.4.3 Finding intersections $\mathcal{C}$

For a line that passes through points $(x_1,y_1),(x_2,y_2)$, and a line that passes through points $(x_3,y_3),(x_4,y_4)$, the intersections (I_x,I_y) are calculated as:

$$\begin{aligned} I_x &= \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_3y_4 - y_3x_4)(x_1 - x_2)}{D}, \\ I_y &= \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (x_3y_4 - y_3x_4)(y_1 - y_2)}{D} \end{aligned} \tag{5.11}$$

$$D = (x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4) \tag{5.12}$$

Utilizing this formula, the intersection point between every line defined by consecutive points in $\mathcal{A}$ and $\mathcal{B}$ is computed. Some of these intersections do not lie on the polygons $\mathcal{A}$ and $\mathcal{B}$. We filter invalid points and only keep the intersections with x coordinates within the range of both two pairs of points in $\mathcal{C}$. x must satisfy $x_1 \leq I_x \leq x_2$ and $x_3 \leq I_x \leq x_4$. (Restrictions on y coordinates are automatically satisfied if x coordinates are within the correct range.)

5.4.4 Finding points $\mathcal{A}$ inside $\mathcal{B}$

For a convex polygon, each edge is assigned a direction in clockwise order, as in Figure 5.3. Then if and only if a point (x,y) lies on the same side of all the edges of the polygon, it lies inside the polygon. Let the endpoints of the line segment be (x_1,y_1) and (x_2,y_2). We compute:

$$(y - y_1)(x_2 - x_1) - (x - x_1)(y_2 - y_1) \tag{5.13}$$

where a positive result means that the point lies on the left of the line, a negative result means that the point lies on the right of the line, and zero means that the point lies on the line. Each point in $\mathcal{A}$ is checked against the line segments defined by $\mathcal{B}$ to determine $\mathcal{A}_B$, the set of points in $\mathcal{A}$ lie within $\mathcal{B}$, and the opposite is done to determine $\mathcal{B}_A$. The full set of points defining the intersection of $\mathcal{A}$ and $\mathcal{B}$ is then defined by $\mathcal{I} = \mathcal{A}_B \cup \mathcal{B}_A \cup \mathcal{C}$.

5.4.5 Calculating the area of a polygon

Let (x_i,y_i) represent the coordinates of the i-th point of a polygon. Let

$$\begin{aligned} x &= [x_1, x_2, ..., x_n]^T, x^* = [x_2, x_3, ..., x_n, x_1]^T, \\ y &= [y_1, y_2, ..., y_n]^T, y^* = [y_2, y_3, ..., y_n, y_1]^T \end{aligned} \tag{5.14}$$

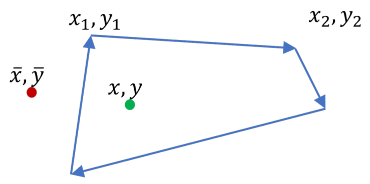


Figure 5.3: A polygon with edges marked with directions in clockwise order. Relative to each line segment, the point (x, y) lies on the right side, while $(\bar{x}, \bar{y})$ lies on the left of one line segment and on the right of three line segments. Credit: Xinxuan Lu.

The area of a polygon is computed by:

$$Area = \frac{1}{2}(x^T y^* - y^T x^*) \tag{5.15}$$

The areas of $\mathcal{I}$, $\mathcal{A}$, and $\mathcal{B}$ are thus computed, and equation 5.10 is used to determine the PIoU.

5.4.6 Batched implementation

Polygon IoU loss is applicable for convex polygons with any number of corners. However, a variable number of points among polygons impedes calculation in a batched, vectorized implementation. Thus, for batched implementation, we restrict inputs to a fixed number of points per batch (in practice, for 3D bounding box calculation, all polygons will be four-sided). For 4-sided polygons, there are at most 8 points in the set $\mathcal{C}$, at most 4 interior points in each set $\mathcal{A}_B$ and $\mathcal{B}_A$, and at most 8 points in $\mathcal{I}$. So, the size of the vector that represents the intersection region is set to 8. If the actual number of points in the set $\mathcal{I}$ is less than 8, the set is padded with repeated points which won't alter the result of (5.15).

For polygons with P points, the maximum number of corner points in $\mathcal{I}$ is set to be $2P$. The batched implementation of a function empirically computes forward and backward training passes significantly faster than a non-batched loop-based implementation of PIoU. Pseudo-code for a batched, vectorized implementation of PIoU with batch size B is given below. The shape of the output tensor at each step is given in square brackets.

```
Algorithm: Batched Tensor PIoU
Inputs: $polygonA$ and $polygonB$. Each $[B,P,2]$.
▷ $\mathcal{C}$ = intersections of all line segments in $polygonA$ and $polygonB$ using (0, 0) to fill the empty. $[B,2P,2]$.
▷ $\mathcal{A}_B$ = all points of $polygonA$ that are inside $polygonB$, using (0, 0) to fill the empties. $[B,P,2]$.
▷ $\mathcal{B}_A$ = all points of $polygonB$ that are inside $polygonA$ using (0, 0) to fill the empties. Tensor shape: $[B,P,2]$.
▷ $overlap$ = union of $[\mathcal{C}, \mathcal{A}_B, \mathcal{B}_A]$. $[B,4P,2]$.
▷ sort $overlap$ in decreasing order with respect to the distance from (0, 0)
▷ keep the first 8 points in $overlap$. $[B,2P,2]$.
▷ $placeholder$ = the points farthest from (0, 0) in $overlap$. $[B,1,2]$.
▷ replace (0, 0) in $overlap$ with $placeholder$
▷ $areaO$,$areaA$,$areaB$ = areas of $overlap$, $polygonA$, and $polygonB$. Each $[B]$.
▷ PIoU = $areaO/(areaA + areaB - areaO)$. $[B]$.
```

5.4.7 Edge Cases

When an edge from polygon A and an edge from polygon B are parallel, the intersections between the two edges are ill-defined because the denominator in (5.11) approaches zero. This occurs when: i.) two edges coincide with the same line. ii.) Two edges are parallel but not coincident with the same line. In case i.), the points of $\mathcal{A}$ and $\mathcal{B}$ already suitably define the intersection points, so including points of $\mathcal{A}$ exactly on an edge of $\mathcal{B}$ in $\mathcal{A}_B$ covers this case. In case ii.), the two edges have no intersection. Thus, we can remove these intersections from $\mathcal{C}$ for numerical stability.

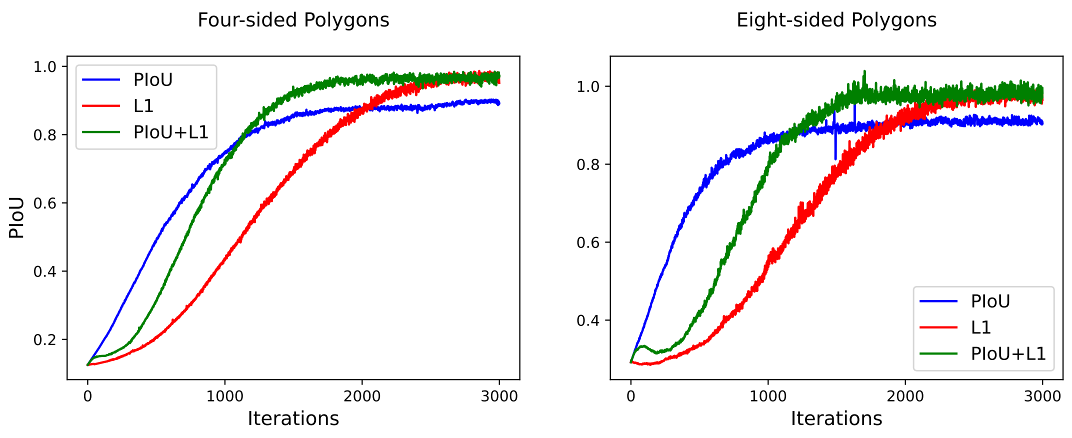


Figure 5.4: PIoU score versus iteration for (left) 4-sided simulated polygons and (right) 8-sided unrestricted simulated polygons. Credit: Xinxuan Lu.

5.5 Experiments on simulated polygons

5.5.1 Four-sided convex polygons

We generate two sets of quadrilaterals (4-sided polygons), with one set as the initial polygons and one set as the ground truth. The polygons are generated as centers and offsets of four points to ensure they are convex. We use the Adam optimizer to regress predicted polygons with the goal of approximating the ground truth polygons. The polygons are generated in a batch of 32 for the training. We compare the result of L1 loss, PIoU loss, and a combination of L1 and PIoU loss. The IoU with respect to iterations is plotted on the left of Figure 5.4. The results take the average of 5 independent trials. The PIoU loss converges the fastest in the beginning. However, the PIoU loss does not achieve a high IoU when it converges. PIoU+L1 loss has the fastest convergence speed and accuracy after around 2000 iterations.

5.5.2 Eight-sided unrestricted polygons

We repeat this experiment with 8-sided polygons, this time not restricting the predicted polygons to be convex. IoU versus optimization iteration is plotted on the right of Figure 5.4. The results take the average of 5 independent trials. The results are similar to the 4-sided polygon case. The PIoU loss converges the fastest initially, while PIoU+L1 loss converges faster than L1 loss alone and additionally reaches the highest overall IoU score. Non-convex polygons produce slightly more noise in loss curves, visible in the PIOU and PIOU+L1 curves of Figure 5.4 (right). This experiment on simulated 8-sided polygons shows that our PIoU loss has good performance even when the polygons are not convex and have more than four sides.

5.5.3 Computation speed

We compare the computation speed of our batched implementation of PIoU loss relative to a pixel-wise IoU loss (as used for object segmentation tasks). For a batch size of 1, our PIoU loss is 4.0x faster than the pixel-wise implementation. PIoU loss is 56.0x faster for a batch size of 16 and 281.6x faster for a batch size of 128.

5.6 Experiments on KITTI 3D

PIoU loss supports polygons with any number of points. However, there are not existing detection problems well suited to evaluating polygons with more than 4 points. We test it on 3D detection problems where the projection of 3D bounding boxes to the image plane can be separated into two quadrilaterals, the front 4 and back 4 corner coordinates of the 3D bounding box. We incorporate PIoU loss into RTM3D [152], MonoCon [153], and MonoRCNN [151] and test each on the KITTI 3D benchmark [160]. In all three models, we compare the performance of using PIoU+L1 loss and only using L1 loss. The rest of this section describes

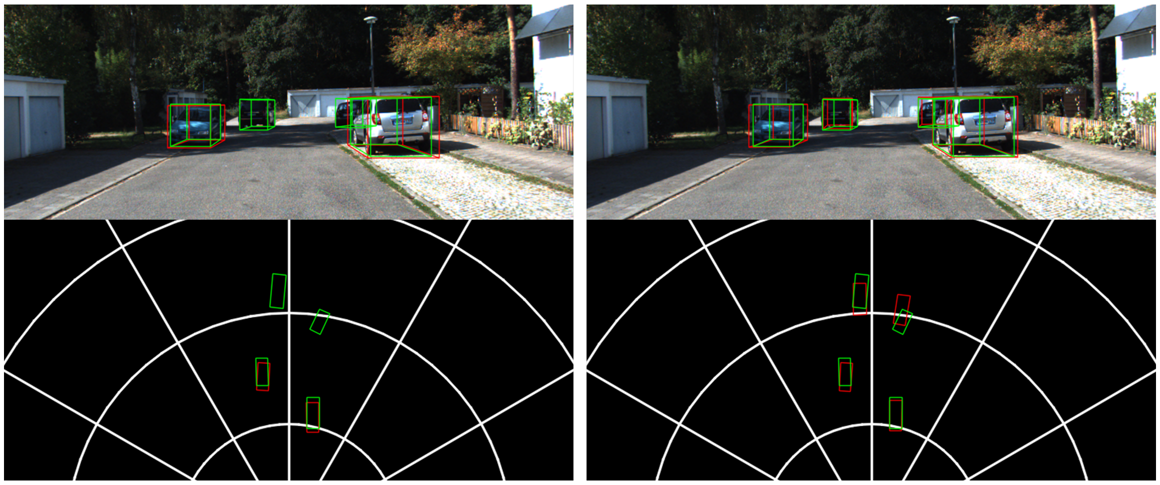


Figure 5.5: Predicted (red) and ground truth (green) 3D bounding boxes from the train/val split of KITTI dataset from MonoCon object detection model trained with L1 loss (left) and PIoU+L1 loss (right). Top images show the predicted bounding boxes in 3D space, and bottom images show the corresponding predicted footprints in a birds-eye view. Credit: Xinxuan Lu.

the models, modifications, and experimental parameters for each experiment. The results are listed in Section 5.7.

5.6.1 Dataset

We use KITTI 3D object detection benchmark as our training and evaluation dataset. As KITTI does not allow more than three submissions and the labels for the testing set are not released, we follow [153] to divide the official training set of KITTI 3D into 3712 training images and 3769 evaluation images. (We use *train/val* to represent this split.) Three classes of objects, cars, pedestrians, and cyclists, are used for training. During training, only left-camera images and ground-truth labels are used (calibration matrices are used only to produce pixel-space 3D keypoint projections.)

5.6.2 Evaluation Metrics

The objects in KITTI are categorized into easy, moderate, and hard according to their height, truncation level, and occlusion ratio. We evaluate the results for each category using the same evaluation guidelines as KITTI. As suggested by KITTI in 2019, we use 40 recall positions to calculate the average precision (AP) of results. KITTI sets different 3D bounding box overlap thresholds for cars (70%) and cyclists (50%). We evaluate cars at 70%, 50% and 30% AP_{3D} and cyclists at 50% and 25% AP_{3D}.

5.6.3 Ensuring convexity

As our polygon IoU loss is accurate when the two polygons are convex, we encourage the predicted keypoints to form convex polygons during training. The ground truth keypoints are projected from 3D cuboids, so it is guaranteed that they form two convex quadrilaterals. We ensure that the initial predictions of the model are convex by adding a small offset (4 corner points of a square centered at the origin) to the predicted keypoints at initialization. During our training and testing, we find that a convex initial prediction is sufficient to make the PIoU loss converge.

5.6.4 RTM3D

5.6.4.1 Experiment Settings

We modify an unofficial implementation [340] of RTM3D to do our experiments. We do not follow RTM3D to assume a Gaussian kernel for keypoints. We solve a least-squares problem to obtain the best-fitting 3D bounding boxes from the predicted keypoints, 3D dimension, and orientation. For comparison, We add our

Model + Loss Types	AP_{70}			AP_{50}			AP_{30}		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
RTM3D (L1)	3.88	2.55	1.96	24.13	**18.62**	**15.34**	**51.31**	40.04	**35.03**
RTM3D (L1+PIoU)	**4.08**	**2.70**	**2.15**	**25.02**	18.58	15.25	51.10	**41.05**	34.95
MonoCon (L1)	22.27	16.98	14.69	62.00	46.19	**41.71**	85.09	65.94	60.97
MonoCon (L1+PIoU)	**24.93**	**18.45**	**15.49**	**63.88**	**46.92**	41.01	**85.88**	**66.60**	**61.43**
MonoRCNN (L1)	**16.94**	**13.73**	**11.67**	49.87	**38.65**	33.47	**76.57**	**60.93**	**52.63**
MonoRCNN (L1+PIoU)	16.73	13.62	11.63	**51.14**	38.63	**33.50**	74.05	59.99	51.85

Table 5.1: Results for RTM3D, MonoCon, and MonoRCNN on KITTI 3D Car on train/val split, evaluated by AP_{3D} with IoU thresholds of 0.7, 0.5, and 0.3. Best result for each model at each threshold is shown in bold.

PIoU loss to regress the 2D projected keypoints of 3D bounding boxes. We use the same training settings for both the baseline and modified model. We use an Adam optimizer with an initial learning rate of 0.0002 which decreases by a factor of 0.1 in epochs 150 and 180. The weight decay is 1e-6. We train for a total of 200 epochs with a batch size of 16. The experiments run on Ubuntu 20.04 and RTX A5000. It took 14.3 hours to train the model with PIoU loss and 11.5 hours to train the model with L1 loss. Adding PIoU loss computation to the RTM3D model only slightly increases the training time.

5.6.4.2 The Least-Squares Problem

We define the least-squares problem for finding the best-fitting 3D bounding boxes similar to the definitions in RTM3D. For each predicted object, $\hat{P}$ represents the eight corner points of the 3D bounding box on the 2D image plane. $\hat{\Theta}, \hat{D} = [\hat{h}, \hat{w}, \hat{l}]^T, \hat{d}$ represent the predicted orientation, dimensions, and depth of the 3D bounding box. $T = [x, z, d]^T$ represents the position of the 3D bounding box, where d represents the horizontal depth. $f(D, T, \Theta)$ maps the 3D bounding box to the 8 corner points in the image plane. We set $\alpha_P = 0.05, \alpha_T = 1, \alpha_D = 1, \alpha_d = 1$ in our experiments. The least-squares problem is defined as

$$\begin{aligned} \max_{D,T,\Theta} \alpha_P \|f(D,T,\Theta) - \hat{P}\|^2 + \alpha_T \|\Theta - \hat{\Theta}\|^2 \\ + \alpha_D \|D - \hat{D}\|^2 + \alpha_d \|d - \hat{d}\|^2 \end{aligned} \tag{5.16}$$

5.6.5 MonoCon

5.6.5.1 Experiment Settings

We use an unofficial implementation [341] of MonoCon. For comparison, we add our polygon IoU loss to regress the 2D projected keypoints of 3D bounding boxes. We use the same training settings for both the baseline and modified model. The batch size is 8, and the total epoch number is 240. Following the original paper, We use an AdamW optimizer with a weight decay of 0.00001. We use a cyclic learning rate scheduler with an initial learning rate of 0.000225 which first gradually increases to 0.00225 with a step ratio of 0.4 and then gradually drops to $2.25e-8$. The experiments run on Ubuntu 20.04 and RTX A5000. Figure 5.5 shows predicted outputs from the implemented model.

5.6.6 MonoRCNN

5.6.6.1 Experiment Settings

We modify the official code of MonoRCNN to incorporate our polygon IoU loss and use the original code to train the baseline. For comparison, we add PIoU loss to regress the 2D projected keypoints of 3D bounding boxes. We use the same training settings for both models. We train for 60000 iterations with a batch size of 8. The initial learning rate is 0.01 and is reduced by 0.1 after 30k, 40k, and 50k iterations. The experiments run on Ubuntu 20.04 and RTX A5000.

5.7 Results

Sections 5.7.1 and 5.7.2 present the results of each model on the KITTI cars and cyclists data, respectively, and Section 5.7.3 shows the performance changes for each model across various phases of training. Note that all AP scores are relatively low when compared with leading benchmark performance on the KITTI 3D detection dataset as a whole. This is because monocular 3D detection is relatively challenging when compared to 3D detection with sensor fusion as a whole and viewpoint agnostic models additionally cannot incorporate scene information explicitly or implicitly into model structure or learning.

5.7.1 Results on KITTI Cars

Table 5.1 shows the final AP scores on KITTI 3D cars with different IoU thresholds. On RTM3D, PIoU loss has on average modestly better AP scores than the original model. Notably, AP_{70} exclusively improves across the easy, moderate and hard subsets of the data, suggesting that PIoU offers the most benefit at higher IoU thresholds because L1 is often suitable for producing "decent" 3D detection results. Averaged across all difficulties, PIoU results in +0.18% AP_{70}, +0.24% AP_{50}, and +0.25% AP_{30}. Proportional to baseline scores, AP_{70}. achieves the largest relative increase.

For the MonoCon model on KITTI cars, the proposed PIoU + L1 loss gives better results than the original model for nearly all different difficulty levels and at all IoU thresholds. (for the IoU threshold of 0.5, PIoU loss gives better results on the easy and moderate difficulty but not on the hard difficulty.) Again, the largest improvements from PIoU loss are seen at higher requisite IoU thresholds (+1.64% AP_{70}, +0.64% AP_{50}, and +0.64% AP_{30}).

Lastly, the performance of the MonoRCNN model on KITTI cars is marginally worse when trained with PIoU + L1 loss versus L1 loss alone. However, there is still improvement at some IoU thresholds (-0.12% AP_{70}, +0.42% AP_{50}, and -1.42% AP_{30} averaged across difficulty levels), and the general trend across models holds that PIoU + L1 loss yields greater relative gain at stricter IoU thresholds.

Across all models, PIoU loss yields more improvements on easier difficulty levels. This trend is intuitive, as the ability of a loss function to improve performance is limited when features relevant to an object are not visible or are highly obscured; better model architectures can likely yield more performance improvement in these cases.

Model	AP_{50}			AP_{25}		
	Easy	Mod	Hard	Easy	Mod	Hard
RTM3D	0.03	0.03	0.03	3.44	2.00	1.56
RTM3D*	**0.04**	**0.04**	**0.04**	**5.97**	**3.17**	**2.81**
MonoCon	4.25	1.96	1.90	**19.20**	**10.84**	**10.22**
MonoCon*	**4.69**	**2.63**	**2.19**	18.27	10.22	9.53
MonoRCNN	3.05	2.01	2.04	15.98	9.05	9.14
MonoRCNN*	**4.31**	**2.73**	**2.55**	**19.05**	**11.22**	**11.28**

Table 5.2: Evaluation results for RTM3D, MonoCon, and MonoRCNN on KITTI 3D Cyclist on the train/val split, evaluated by 3D *AP* with IoU thresholds of 0.5 and 0.25. A * indicates models are trained with PIoU+L1 loss; other models are trained with L1 loss. Best result for each model at each threshold is shown in bold.

Model + Loss Types	AP_{50} @ 200 epochs			AP_{50} @ 160 epochs			AP_{50} @ 100 epochs		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
RTM3D (L1)	24.13	**18.62**	**15.34**	24.29	**18.74**	**15.35**	17.66	13.80	11.15
RTM3D (L1+PIoU)	**25.02**	18.58	15.25	**24.91**	18.59	15.30	**20.41**	**15.11**	**13.22**
Improvements	0.89	-0.04	-0.09	0.62	-0.15	-0.05	2.75	1.31	2.07

Table 5.3: Evaluation results for RTM3D on KITTI 3D Car on the train/val split, evaluated by AP_{3D} with an IoU threshold of 0.5 at epoch 100, 160, and 200. Best result at each epoch is shown in bold.

Model + Loss Types	AP_{70} @ 240 epochs			AP_{70} @ 160 epochs			AP_{70} @ 80 epochs		
	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
MonoCon (L1)	22.27	16.98	14.69	19.45	14.69	12.27	3.47	3.08	2.45
MonoCon (L1+PIoU)	**24.93**	**18.45**	**15.49**	**19.94**	**15.35**	**12.98**	**14.84**	**10.84**	**8.86**
Improvements	2.66	1.47	0.8	0.49	0.66	0.71	11.37	7.76	6.41

Table 5.4: Evaluation results for MonoCon on KITTI 3D Car on the train/val split, evaluated by AP_{3D} with an IoU threshold of 0.7 at epoch 80, 160, and 240. Best result at each epoch is shown in bold.

5.7.2 Results on KITTI Cyclists

Table 5.2 compares the AP scores of incorporating PIoU loss versus L1 loss alone on KITTI 3D cyclists. For RTM3D, the accuracy is very low and therefore the change in prediction accuracy between the loss functions is negligible when IoU thresholds are 0.5., but PIoU+L1 loss significantly improves performance over L1 loss alone for an IoU threshold of 0.25 (+1.65% AP_{30} averaged across all difficulties.)

MonoCon performance is less notable on cyclists than on cars. Still, the performance of a model trained with PIoU+L1 loss is better than L1 alone at the more stringent IoU threshold of 0.5 (+0.56% AP_{50}.)

For MonoRCNN, adding PIoU loss strictly improves cyclist detection performance. The AP scores in all IoU thresholds and difficulty levels increase by around 1-3%. Here again, the largest improvements in prediction accuracy occur for the easy subsets of the data (+1.26% AP_{50} and +3.07% AP_{25}).

5.7.3 Model Performance at Different Epochs

Lastly, we test the convergence speed of PIoU + L1 loss versus the baseline L1 loss. Table 5.3 shows the AP scores for RTM3D on KITTI cars at different epochs. When using PIoU loss, the AP scores early in training are significantly better than the baseline model (e.g. +2.75% AP_{50} on easy subset). (A similar trend is visible for AP_{70} but this table is omitted for brevity). Table 5.4 similarly shows PIoU+L1 loss results in faster model convergence than the L1 baseline at an IoU threshold of 0.7 for KITTI cars. After 80 epochs, PIoU results in at least +5% AP_{70} for each difficulty, with the largest performance improvement on the easy subset (+11.37%). The results indicate strongly that PIoU+L1 loss converges faster than L1 loss alone.

5.8 Conclusions and Future Work

In this work, we propose an efficient way to calculate IoU between convex polygons with irregular shapes. The batched implementation of the proposed PIoU loss in PyTorch is differentiable and can be used as the loss function for large object detection models. We show that PIoU loss can speed training convergence, both for simulated polygons and on the KITTI 3D detection dataset. We also show that using PIoU+L1 loss can increase the AP scores over L1 loss alone. Improvements vary when we incorporate PIoU loss into different 3D object detection models, with The CenterNet-based models benefitting more than the R-CNN-based model and a best result of +1.64% AP_{70} for MonoCon on KITTI cars. The most notable performance gains occur for highly visible vehicles when a strict IoU metric is required, meaning PIoU is especially helpful in transforming “good” predictions to “great” predictions.

This work tests PIoU loss on 3 different 3D object detection models for a benchmark dataset with relatively constrained (4-sided) polygons. In future work, we would like to incorporate the loss function into more difficult cases where rectangular bounding boxes are not sufficiently expressive and a detection formulation is preferred to a segmentation-based model formulation, providing an “in-between” for expressive detections of middling complexity. The results of PIoU in this work are quite promising, and we hope that future work can test the loss formulation on a larger sampling of methods and datasets, and in combination with other loss formulations.

6. A Multi-Camera Vehicle Tracking Baseline

6.1 Introduction

In recent years, 3D detection and tracking datasets in the autonomous vehicle domain have led to marked advancements in perception and planning algorithms and AV technology more generally [124, 284, 285]. But designing autonomous technologies from an ego-vehicle perspective alone is not enough. Studies have shown that control algorithms designed for an individual vehicle's objectives can cause rippling instabilities in traffic [16], while controllers designed with global traffic objectives in mind can significantly reduce congestion [10, 342].

Automatic traffic monitoring offers a tremendous but under-exploited opportunity to address this issue. Computer vision research has progressed sufficiently in other fields such that efficient algorithms for traffic monitoring at scale likely exist, and state and federal transportation agencies maintain camera networks with tens of thousands of cameras nationally; increasingly ubiquitous edge sensing devices only add to the number of potentially useful traffic cameras. Moreover, in several cases, multi-camera systems have been deployed at considerable scale specifically to study the effects of *intelligent transportation systems* (ITS) and AVs on traffic [34, 70, 79, 80, 343]. Similarly, work on autonomous management of city-scale traffic will benefit immensely from the ability to track vehicle movements precisely (often requiring 3D detection) across many cameras [296]. It yet remains to be explored whether existing algorithms can achieve tracking performance suitable for fine-grained traffic analysis (i.e. HOTA above 75% and over 95% mostly tracked objects), where small localization errors or a single ID switch can be damaging in understanding a scenario [50].

We seek to enable research on precise vehicle tracking in the traffic monitoring context, with emphasis on the challenges of multi-camera tracking faced in systems such as [34, 70, 79, 80, 343]. Work in this field has been slowed by a lack of 3D multi-camera tracking data; this work addresses this shortage to enable development and evaluation of tracking methods to meet the needs of the next generation of intelligent traffic systems and AV research. [1]

The primary contribution of this work is the introduction of a novel dataset suitable for multi-camera tracking, consisting of 877,000 3D vehicle bounding boxes annotated across 16-17 cameras with dense viewpoints covering 2000 feet of interstate roadway near Nashville, TN. The *Interstate-24-3D Dataset* (I24-3D) introduced in this work is comprised of 3 *scenes* (sets of videos recorded at the same time from different cameras), recorded at 4K resolution and 30 frames per second. Vehicle 3D bounding boxes are annotated by hand for 720 unique vehicles. I24-3D is the first 3D multiple-camera dataset in a traffic monitoring context with real videos and tracks objects across a larger set of cameras than any other multi-camera tracking dataset. **The secondary contribution of this work is the benchmarking of a number of existing algorithm combinations to assess the difficulty of 3D multi-camera tracking on this dataset,** with results (best performance of 44.8% HOTA and 62% mostly tracked objects) showing that the implemented methods achieve good performance but do not produce data suitable for fine-grained traffic analysis.

The rest of this chapter is organized as follows: Section 6.2 describes the data and annotations included in I24-3D. Section 6.3 provides details of benchmarking experiments using the dataset, and Section 6.4 describes the results. Additional details on the dataset including annotation details, accuracy metrics, timestamp synchronization efforts, additional experimental settings and implementation details, unabridged results, and privacy considerations are included in Appendices I - N.

6.2 The I24-3D Dataset

This section introduces the I24-3D Dataset, detailing the location of the cameras, describing the annotations, vehicle classes, and suitable uses, and providing annotation quality metrics. Data is available at i24motion.org, and code demonstrating usage is available at github.com/DerekGloudemans/I24-3D-dataset.

[1] This chapter is repurposed from [95].

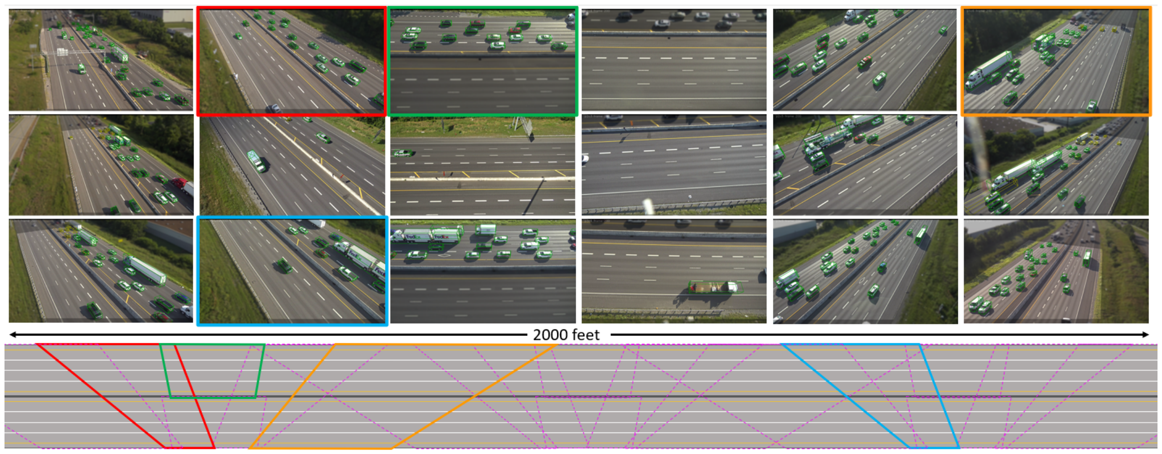


Figure 6.1: Example annotated (green boxes) frames from each camera field of view for one scene of the I24-3D Dataset. The approximate field of view for each camera is shown on the overhead roadway diagram below (some cameras shown in unique colors as examples). Regions outside of the considered field of view for each camera are blurred for this visualization. Cameras provide coverage of 2000 feet of Interstate-24 near Nashville, TN.

6.2.1 Overview

The I24-3D Dataset consists of 3 *scenes*, or collections of video data recorded simultaneously from 16-17 cameras, densely covering a section of roughly 2000 feet of roadway. Each scene is 60-90 seconds long, recorded at 4K resolution and 30 frames per second, and features manually annotated 3D bounding boxes on every vehicle visible within the field of view of each camera suitable for vehicle re-identification, 3D object detection, tracking, and multi-camera tracking tasks (see Table 6.1.) Over 275 person hours were spent annotating the data.

6.2.2 Location

I24-3D was recorded using I-24 MOTION [34], an open-road testbed along Interstate 24 near Nashville, Tennessee. The utilized portion of this testbed contains 18 cameras mounted on three 110-foot tall roadside poles, spaced at roughly 500 feet and covering an approximately 2000 foot field of view on the interstate [85]. (Due to periodic camera outages, each scene contains footage from only 16-17 cameras).

6.2.3 Annotation Description

Annotations are provided in a roadway-aligned coordinate plane, where x-coordinate indicates distance along the roadway and y-coordinate indicates lateral (lane) position, of the bottom center rear of the vehicle. For each direction of travel in each camera field of view, a *homography* relates the roadway coordinate system to the pixel coordinates of the field of view. We rely on standard perspective transforms [344] for this conversion (see Appendix I), assuming the roadway visible in each field of view can be reasonably represented by a flat plane with a relevant *field of view* (FOV) comprising most of each image (masks are provided for regions

Scene	Time (s)	Cameras	Frames	Boxes	IDs	VMT	Description
1	90	17	45900	291k	324	118	Free-flow traffic
2	60	16	30600	146k	114	24.4	Slow traffic, snow conditions
3	60	16	28800	440k	282	67.0	Congested traffic
Total	210	-	105300	877k	720	209	-

Table 6.1: Summary of scene data for I24-3D dataset. *Time* indicates the total global duration of a scene (each video segment for the scene has that duration). *Frame* count is aggregated across all cameras in the scene, *cameras* indicates the number of active cameras for the scene. *Boxes* indicates number of 3D bounding boxes, *IDs* indicates unique vehicle trajectories.

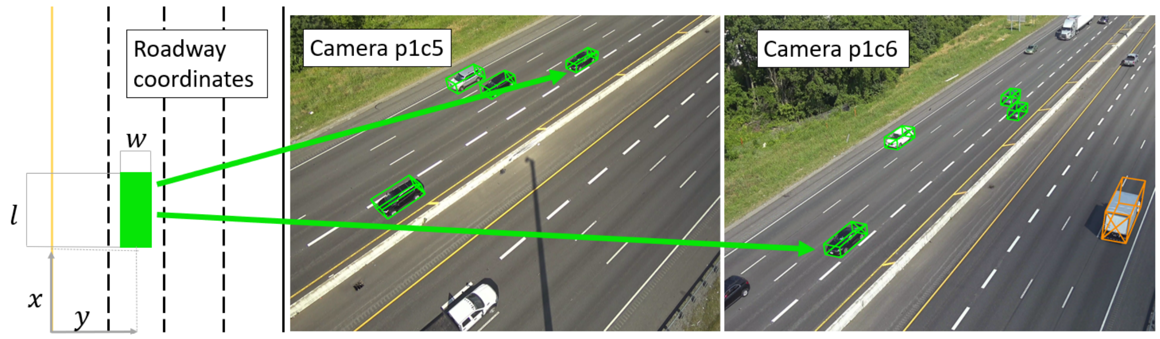


Figure 6.2: Example single annotation. The annotation is stored in roadway coordinates (left) but can be projected into cameras 5 and 6 on pole 1 (p1c5 and p1c6).

falling outside of the FOV for each camera). All distances are given feet, as the geometry of the roadway is laid out in feet (e.g. lanes are 12 feet wide).

A single vehicle 3D bounding box annotation includes *vehicle class*, unique *vehicle ID*, bounding box *length*, *width*, and *height* (fixed for all annotations for a single vehicle), *vehicle roadway position*, *originating camera*, *timestamp*, and *frame index*. This information is sufficient to losslessly project the annotation into the originating camera, or into any other camera in which it is visible. Figure 6.2 shows an example. **Object localization is precise, with 1.24 ft average positional error between annotations of the same vehicle labeled in multiple cameras, and 0.5 ft average dimensional error**. (See Appendix J and K.)

6.2.4 Vehicle Classes

Vehicles are classified into six classes: *sedan*, *midsize* (minivan, SUV or compact SUV), *van*, *pickup*, *semi* (tractor-trailer), or *truck*). Figure 6.3 depicts example annotations for each class as well as the total number of annotations for each class. We make one additional distinction: vehicles other than semis that tow trailers are classified with the towing vehicle's class, but bounding boxes are drawn to include the trailer. This choice reflects that a vehicle and trailer behave as a single semi-rigid body. Vehicle IDs with trailers include: Scene 1: [288, 133, 7, 138, 43, 270, 245, 216], Scene 3: [225, 105, 15, 148, 247, 219].

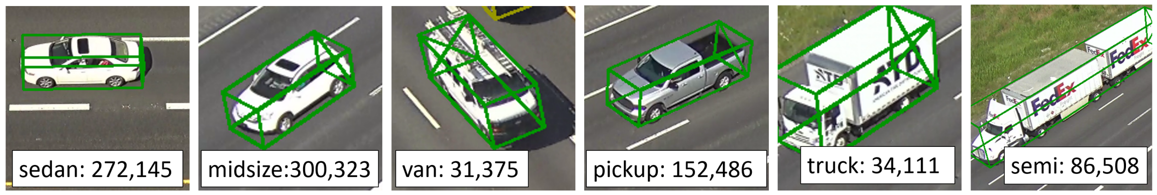


Figure 6.3: Example vehicles and vehicle class annotation counts for the I24-3D dataset.

6.2.5 Dataset Uses and Comparison

The I24-3D dataset provides annotations of sufficient richness for a variety of canonical computer vision problems, including object reidentification, 2D and 3D detection and tracking. Most notably, multiple videos from a single scene can be used for multiple-camera tracking tasks, and the presence of 3D labels in this dataset enables explicit modeling of a shared 3D space for object tracking. Table 6.2 provides a comparison of the suitable uses of the I24-3D dataset and the most similar existing datasets. Notably, I24-3D is the only dataset in a traffic monitoring context that allows for 3D multi-camera tracking.

6.3 Benchmarking Experiments

To provide an initial gauge of tracking difficulty and existing algorithm performance on I24-3D, we benchmark a set of tracking methods on this dataset. Experimental protocol, metrics for evaluation, and implemented algorithms are briefly described in this section.

Dataset	Resolution	Detection		MOT		MCT		Boxes	Frames	Cameras
		2D	3D	2D	3D	2D	3D			
WILDTRACK [295]	1920×1080	✓		✓		✓	✓	38k	61k	7
KITTI [284]	1382×512	✓	✓	✓	✓			200k	15k	1
NuScenes [285]	1600×900	✓	✓	✓	✓	✓	✓	12M	40k	6
BoxCars116k [115]	*varies*	✓	✓					116k	116k	1
UA-DETRAC [215]	1920×1080	✓		✓				1.2M	140k	1
CityFlow [296]	960×540	✓		✓		✓		229k	117k	25*
Synthehicle [298]	1920×1080	✓	✓	✓	✓	✓	✓	4.62M	6.7k	7
I24-3D (Ours)	3840×2160	✓	✓	✓	✓	✓	✓	877k	105k	16-17

Table 6.2: Suitable uses and metrics for comparable MOT and 3D vehicle detection datasets, grouped by traffic monitoring (bottom) and other contexts (top). *MOT* indicates multiple object tracking (in 2D or 3D), and *MCT* indicates multiple camera tracking (with 3D tracking requiring a unified tracking space). *Boxes* indicates the total number of monocular bounding box view annotations, *Frames* indicates the total number of annotated frames in the dataset, *Cameras* indicates the number of camera views in a single scene. A * indicates some camera fields of view have large gaps between them (non-overlapping).

6.3.1 Experimental Protocol

Each scene is split into temporally contiguous training and validation partitions (the first 80% and the last 20% of each scene, respectively). Training is performed exclusively using the training partition. All training is performed locally on RTX6000 GPUs, and detection models are trained until convergence. During tracking we maintain tight 1/60th second synchronization between each video using corrected frame timestamps (see Appendix J), skipping frames as necessary to maintain a 15 Hz nominal frame rate.

For tracking evaluation, we find a best-fit 3rd order polynomial spline for each ground truth vehicle to obtain a continuous object representation in roadway coordinates. Predicted vehicle trajectories are compared against boxes sampled from the best-fit spline for each object. We linearly interpolate between the spline-sampled boxes and the tracker-output predictions at 30Hz to produce object sets at the same discrete times. Additional experimental details are given in Appendix L.

6.3.2 Metrics

We compare tracker performance using the clearMOT metrics [216], the MT/ML metrics used in [345], and HOTA [218]. We also consider the percentage of ground truth (*GT%*) and predicted objects (*Pred%*) matched to at least one predicted or ground truth object, respectively). To account for time-synchronization errors, we use a requisite 30% 2D-footprint IOU threshold between predicted and ground truth objects.

6.3.3 Algorithms Implemented

A variety of multi-camera 3D MOT pipelines are assembled, each requiring 3 algorithmic components: i.) a 3D object detector, ii.) an object tracker / association method, and iii.) a method for combining objects across cameras. We briefly describe algorithms implemented for each stage (implementation and parameter details can can be found in Appendix L).

3D Detectors:

- **Monocular 3D Detector (Single3D)** - a Retinanet model with Resnet34-FPN backbone [132]. The formulation is camera-agnostic (as training a separate model for each camera FOV is infeasible both from data scarcity and scalability standpoints.) *Average Precision* (AP) scores for this detector: $AP_{30} = 0.718, AP_{50} = 0.598, AP_{70} = 0.254$. (See Appendix L for experimental details.)
- **Monocular 3D Multi-frame Detector (Dual3D)** - Inspired by recent works utilizing multiple frames for detection and tracking [87], we add the previous frame as detection input. AP scores for this detector: $AP_{30} = 0.810, AP_{50} = 0.714, AP_{70} = 0.572$.
- **Monocular 3D Crop Detector (CBT)** - as described in [91], we train a Retinanet Model with Resnet34-FPN backbone for detecting objects in cropped portions of full frames. AP scores for this detector: $AP_{30} = 0.767, AP_{50} = 0.700, AP_{70} = 0.464$.
- **Ground Truth Detections (GT)** - perfect ground-truth detections.

Detector	Tracker	DF	TF	HOTA	MOTA	Rec	Prec	GT%	Pred%	MT	ML	Sw/GT
Crop	Byte	✓	✓	23.6	21.3	53.4	64.0	90.5	72.9	25.6	25.0	1.1
Crop	KIOU	✓	✓	24.6	21.4	54.4	64.2	90.5	71.2	27.6	22.3	1.1
Dual3D	Byte	✓	✓	30.9	50.0	65.6	81.9	90.6	93.4	35.9	15.0	0.9
Dual3D	KIOU	✓	✓	39.7	71.6	76.5	93.7	91.5	95.3	52.5	10.4	0.7
Single3D	Byte	✓	✓	27.5	49.3	62.8	83.9	92.1	91.8	29.7	15.4	0.9
Single3D	KIOU	✓	✓	39.9	71.6	76.3	94.1	93.4	**95.6**	51.6	8.8	0.7
Crop	Byte		✓	15.2	-16.5	43.5	47.0	90.4	59.8	14.5	32.2	1.5
Crop	KIOU		✓	20.7	-2.1	51.6	52.9	90.2	53.7	25.5	26.4	1.5
Dual3D	Byte		✓	38.7	75.0	80.2	93.8	93.0	93.6	59.0	8.0	0.8
Dual3D	KIOU		✓	**44.8**	77.0	**83.0**	93.2	91.7	92.3	**63.8**	8.8	**0.5**
Single3D	Byte		✓	38.2	72.6	80.6	90.9	**94.9**	92.5	58.7	**4.7**	1.1
Single3D	KIOU		✓	**44.8**	**77.1**	**83.0**	93.4	93.3	91.3	62.2	7.8	**0.5**
Crop	Byte	✓		19.2	34.7	58.3	72.8	91.9	81.4	27.1	16.6	2.4
Crop	KIOU	✓		19.3	32.1	57.9	71.4	91.9	79.7	25.9	17.3	2.4
Dual3D	Byte	✓		20.9	60.2	64.2	94.8	92.7	94.7	29.5	8.7	2.8
Dual3D	KIOU	✓		21.1	60.4	64.0	95.2	92.6	94.7	29.7	8.9	2.7
Single3D	Byte	✓		21.3	60.3	63.9	95.3	93.8	94.2	27.1	7.5	2.6
Single3D	KIOU	✓		21.4	60.3	63.7	**95.5**	94.2	93.9	26.5	7.5	2.6
Crop	Byte			17.6	18.7	59.8	64.0	91.9	73.0	30.4	15.1	3.0
Crop	KIOU			16.9	10.8	57.5	60.0	91.9	66.0	28.2	18.5	3.2
Dual3D	Byte			15.0	55.1	72.8	81.7	93.2	87.5	42.5	6.8	7.3
Dual3D	KIOU			15.1	55.6	72.7	82.2	93.1	87.8	42.3	7.0	7.3
Single3D	Byte			15.1	54.0	72.3	80.8	94.3	85.9	40.5	5.8	7.2
Single3D	KIOU			15.2	54.4	72.2	81.3	94.5	86.1	39.4	5.6	7.1

Table 6.3: Tracking results for each multi-camera tracking pipeline. **Sw/GT** indicates object ID switches per ground truth object. Best result for each metric shown in bold.

Object Trackers:

- **Kalman-Filter IOU Tracker (KIOU)** - as described in [166]. We utilize a contant velocity roadway-coordinate Kalman filter for object position prediction.
- **ByteTracker (Byte)** - noting this tracker's state of the art performance on the MOTChallenge benchmarks [217], we utilize the two-stage association method described in [226], using IOU as both primary and secondary matching criterion and utilizing a Kalman filter as suggested by authors.
- **Crop-based Tracking (CBT)** - as proposed in [91], detection on some frames is performed by re-detecting priors in cropped subsets of the overall frame, and object associations are implicit for these frames.
- **Ground Truth Single Camera Tracklets** - perfect single-camera tracklets.

Cross-Camera Rectification Methods:

- **Detection Fusion (DF)** - as preferred in the AV context [285], detections from all cameras are combined online in roadway coordinates and non-maximal supression with a stringent 1% IOU threshold utilized to eliminate overlapping detections.
- **Trajectory Fusion (TF)**- as proposed in [311], single camera tracklets are compared for spatio-temporal overlap offline, stitched together when a matching criteria is met, and refined to optimally describe the observed set of tracked object positions.
- **None** - as a baseline, object tracklets from each camera are output with no fusion.
- **Both (DF+TF)** - Tracking uses detection fusion, and a subsequent trajectory stitching step is performed to deal with remaining object fragmentations.

6.4 Results

Table 6.3 reports results for each of the above implemented pipelines. The best performing pipeline combines Dual3D detection with KIOU tracking and trajectory fusion (HOTA 44.8%). In general, trajectory fusion alone performs best (across otherwise equal run settings) and no cross-camera rectification strategy (baseline) performs worst. While relatively high MOTA scores are achievable at a low 0.3 IOU threshold (77.1% maximum), HOTA scores are still relatively low when compared to top performing algorithms on MOTchallenge and KITTI [217, 284]. This is primarily driven by relatively low localization accuracy, especially for fast moving vehicles (where a 1-frame timing error results in dropping below a 70% threshold for localization accuracy for an otherwise perfect detection.) See Appendix M for an example HOTA plot at varying localization thresholds.

Even the best pipelines miss 5% of ground truth objects entirely (GT%), and track only 64% of objects for 80% of overall duration (MT). This result demonstrates the difficulty of tracking most or all of the vehicles in a traffic scene at the level of granularity and completeness necessary for in-depth traffic analysis. Even utilizing ground truth detections or single camera tracklets cannot fully mitigate these failures. For brevity, pipelines utilizing ground truth inputs are included in Appendix M; the best-performing pipeline utilizing ground truth detections achieves HOTA 59.6%, and the best-performing pipeline utilizing ground-truth single-camera tracklets achieves HOTA 61.6%. This indicates that the cross-camera tracklet rectification problem is difficult even with great single-camera tracklets.

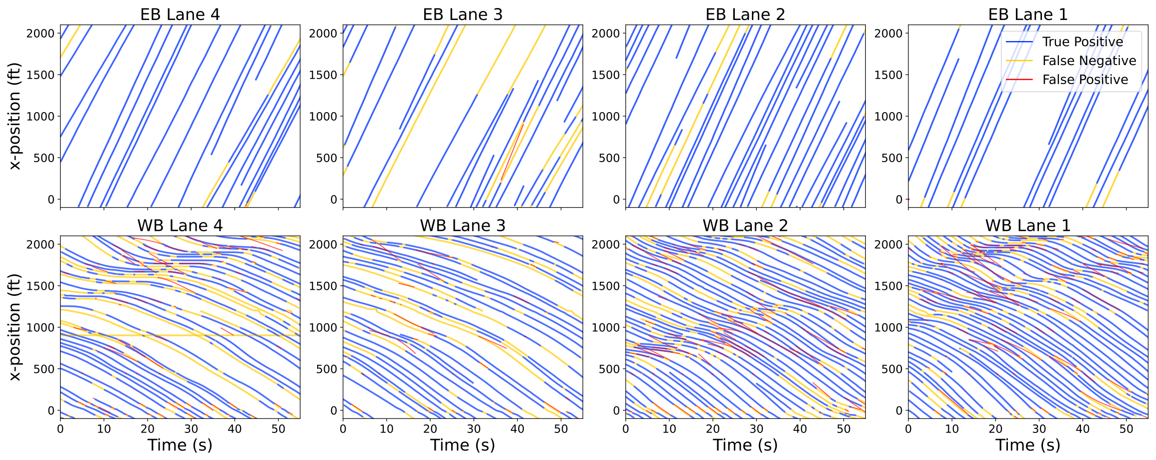


Figure 6.4: Time-space diagrams (object x-position vs time) for each lane for Dual3D +KIOU+TF pipeline on Scene 3 (Lane 4 is rightmost lane in direction of travel). False negatives (yellow), false positives (red) and true positives (blue) shown. In this case, most false positives are closely paired with a false negative, indicating that an object was tracked below the IOU threshold. Lanes farthest from cameras (EB lane 1 and WB lane 4) have the more false negatives in general, likely due to smaller object size and greater object occlusion. In some cases, a predicted object that falls below the IOU threshold with a ground truth object results in a parallel false positive and false negative track.

Table 6.4 reports results for the best pipeline per scene. Scene 1 is easiest across a variety of metrics, with Scene 2 being easier on MOTP (slow-moving objects due to snowy conditions minimizes localization inaccuracies). Per-scene results for all methods are included in Appendix M. Figure 6.4 shows the best performing pipeline's outputs evaluated against ground truth object annotations for Scene 3. Lanes farther from cameras and with high object densities have a much higher rate of false negatives (e.g. westbound (WB) lane 4). Slow-moving, un-occluded objects (e.g. WB Lane 1) are tracked relatively accurately. Faster moving objects (e.g. EB Lane 2) are often tracked, but not accurately enough to surpass the IOU threshold

Scene	HOTA	MOTA	MOTP	Rec	Prec	GT%	Pred%	MT	ML	Sw/GT
1	**58.5**	**89.7**	69.2	**92.9**	**96.7**	**95.3**	**98.4**	**86.3**	**2.2**	**0.02**
2	46.9	77.7	**74.5**	86.2	91.1	90.4	82.4	64.0	9.6	0.49
3	29.1	63.5	64.8	69.9	91.7	89.3	96.1	40.9	14.6	1.05
avg	44.8	77.0	69.5	83.0	93.2	91.7	92.3	63.8	8.8	0.52

Table 6.4: Tracking results for Dual3D + KIOU + TF for each scene. Best score for each metric shown in bold, generally suggesting an easier scene.

requirement. Results on Scene 3 demonstrate the difficulty of tracking all objects in dense stop-and-go traffic, when many objects are occluded for long periods of time.

6.5 Conclusion

This work introduced the I24-3D dataset, a multi-camera 3D vehicle tracking dataset with a total of 57 minutes of video and 877,000 vehicle annotations across 16-17 cameras. It also provided an initial benchmarking of some multi-camera 3D tracking pipelines from existing algorithms, demonstrating the difficulty of tracking on this dataset.

The benchmarking performed in this work represents a first step towards developing and evaluating efficient and accurate 3D multi-camera tracking pipelines. Moreover, though none of the benchmarked pipelines achieved performance suitable for fine-grained traffic analysis (i.e. HOTA ¿ 0.75, mostly tracked objects ¿ 95%), we suspect that there do exist methods or combinations of methods that will perform better than the implemented methods from this work, especially those that better utilize the 3D scene information stemming from multiple cameras in an intensely occlusion-aware manner. We encourage interested researchers to report their results on this benchmark utilizing the protocol described in Section 6.3 and Appendix L. In the future, we look forward to developing such scene-aware MOT methods, armed with a new enabling dataset. We also intend to release a 3D multi-camera tracking challenge with new scenes and cameras from the I-24 MOTION system [34].

7. A Long-Term Vehicle Tracking Baseline

7.1 Introduction

Figure 7.1: Example fields of view from each of the 234 cameras included in the I24V dataset. Each camera is recorded in 1920 $\times$ 1080 resolution and at 30 frames per second. Scene information is provided for each roadway direction of travel in each camera.

Much concerted work has been spent on multiple object tracking benchmarks in recent years, primarily from the perspective of pedestrian tracking in crowds [82, 217] or vehicle tracking from an AV perspective [284, 285]. These datasets generally have high object density, short scenes (1-2 minutes), and short object longevity ($\sim$10 seconds), focusing on high localization accuracy, precision and recall. As a result they do not emphasize challenging aspects of long-term tracking: appearance changes, long-term occlusions, and increasing chance of fragmentation or ID swaps with increasing track length. [1].

Crucially, there is no existing multiple object tracking dataset with a high object density (over 250), long moving object durations (over 5 minutes), and more than 25 overlapping cameras covering a single scene or scenario at the same time. As a result, researchers cannot answer whether existing tracking algorithms are suitable for tracking objects through dense scenes over tens of thousands of frames, because there is no dataset to perform this evaluation on. Such tracking is crucial in the context of traffic science, where origin-destination information for individual vehicles and long-term vehicle behavior are paramount for designing well-fitting models of human driver behavior [37, 54]. It is our goal to provide a video dataset of a different spatial and temporal scale than previous works to enable object tracking research in this vein.

To this end, we present the *Interstate 24 Video* (I24V) dataset. The dataset consists of a single scene, 1 hour in duration, of 4.2 miles of interstate roadway, covered by 234 cameras with overlapping fields of view. Given the scale of this dataset (over 2000 times the video duration of MOTChallenge [82], 500x the duration of KITTI [284] and 80x the scale of CityFlow [296]) traditional manual annotation of objects is infeasible. To combat this difficulty, we provide a set of 270 manually-corrected GPS trajectories from over 100 instrumented vehicles on the roadway during the recording duration. Objects persist for an average of 6.6 minutes (11880 frames average at 30 *frames per second* (FPS)) and a high object density ($>$ 500 across the scene) is typically observable. This annotation set is suitable for assessing object tracking algorithms along recall-oriented metrics. Initial experiments show that existing high-performing trackers fall well short of acceptable tracking performance on data of this scale, and further work is needed to develop suitable algorithms for long-term tracking tasks. We take considerable care to make the data useful for computer vision applications, developing new techniques for keeping camera homographies more accurately aligned

[1]This chapter is adapted from [93]

Dataset	**Cameras**	**Video** (hr)	**Scene** (min)
DukeMTMC [291]	8	11.3	85
Wildtrack [295]	7	1.0	8.6
CityFlow [296]	25	3.3	6.5
Synthehicle [298]	7	17	3
EPFL-Terrace [346]	4	14	3.5
PETS [292]	8	0.2	0.3
pNEUMA Vision [297]	10	3.9	13
I24-3D [95]	17	1.0	1.5
I24-Video (proposed)	234	234	60

Table 7.1: This table summarizes the most comparable existing multi-camera datasets according to *Cameras*, the total number of camera fields of view covering a single scene, *Video*, the total length of all included video, and typical *Scene* duration as estimated from available information for each work.

than existing stabilization methods allow. Succinctly, the contributions of this work are:

1. The largest multi-camera video dataset (234 cameras and 234 hours of video covering a scene with high object density and long object durations).
2. A sparse set of 270 GPS-produced annotations corresponding to 1782 minutes of labeled vehicle trajectory.
3. Preliminary benchmarking of existing object tracking algorithms on this dataset.
4. Precise scene information and a unified curvilinear coordinate system for the entire scene, useful for filter-based tracking and downstream traffic science.
5. Methods for precisely re-aligning camera homographies to account for drift outperforming existing image stabilization techniques in over 99% of cases.

The rest of this chapter is organized as follows: Section 7.2 introduces the dataset, its attributes, and methods used to ensure its fidelity. Section 7.3 describes the numerical experiments and Section 7.4 the results for homography re-estimation methods and for object tracking algorithms benchmarked on the dataset. Much additional explanation and analysis omitted for brevity is available in the Appendices.

7.2 Dataset

This section describes the data released in this work. This dataset includes: i.) 234 hours of video concurrently recorded from 234 cameras. ii.) Scene information for each roadway direction of travel in each camera. iii.) A unified curvilinear coordinate system aligned with the primary roadway direction of travel. iv.) Ground truth GPS trajectories for 270 vehicle runs through the camera fields of view v.) Object detections produced at 30Hz on the video scene. Each is described in more detail in the following sections.

7.2.1 Video Data

7.2.1.1 Location

Video of a single complex traffic scene was recorded using the I-24 MOTION traffic testbed [34]. Briefly, this system is comprised of 294 IP pan-tilt-zoom cameras densely covering a 4.2 mile stretch of 8-10 lane interstate roadway near Nashville, Tennessee. The main system features 40 $\sim$110-foot tall traffic poles, each with six cameras mounted to provide seamless coverage of roughly 500 feet of the interstate. The primary goal of this camera system is to provide accurate, anonymized vehicle trajectory and dimension information to enable traffic science. See [34] for more details. Figure 7.2 provides an overview of system features and a typical camera coverage layout for a single camera pole. Due to the layout of the cameras, any object passing through the whole system is visible in a minimum of 185 cameras, and roughly 1-3 cameras at any point in time with a few exceptions for overpasses and camera pole outages.

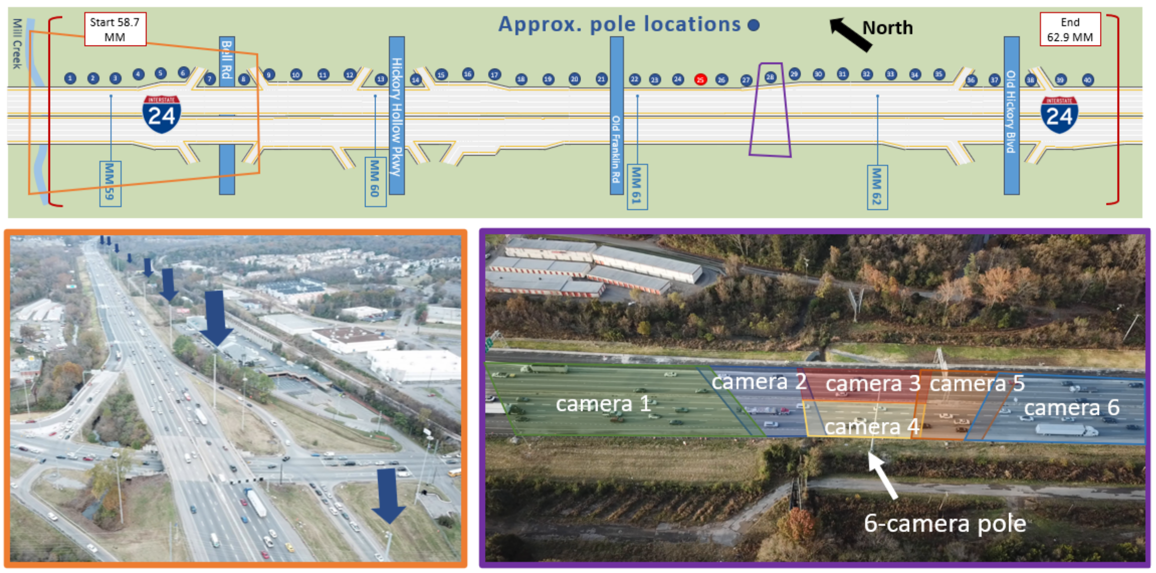


Figure 7.2: **(top)** Graphical overview of the I-24 MOTION system. Each blue dot represents a camera pole with 6 cameras. Red dot indicates a camera pole outage (Pole 25). **(orange)** drone image showing 8 of the 40 system camera poles. **(purple)** Typical 6 camera per pole coverage layout. Best viewed zoomed-in.

7.2.1.2 Recording Details

On a morning in November 2022, video data was recorded from 234 of the 296 cameras simultaneously from 6:00AM to 10:30AM, roughly covering the morning rush hours. The 7:00-8:00AM hour is published here. HD video (1920 $\times$ 1080 pixels) was recorded at 30 frames per second from each of the cameras and stored in H.264 compressed format, totaling $\sim$1 TB. As in [296], any visible license plates are redacted using [347]. Each video is then manually inspected, to remove any pedestrians, private property, or other personally identifiable information (see Appendix T). The one-hour scene has notable features, including i.) several anomalous events, including at least 10 stopped vehicles, ii.) high object density (>500 objects present at most times during the recording), and iii.) significant occlusion of vehicles by taller vehicles with moderate frequency.

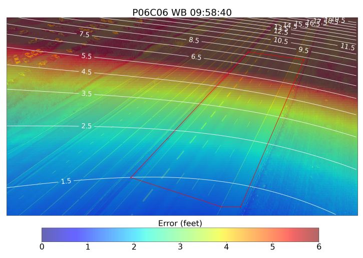
7.2.2 Scene Homography

Particular care with scene information is taken in this work as an accurate transformation from image pixel space to a unified coordinate system is a vital pre-requisite for precise multi-camera tracking. The standard approach [344] utilizes a *homography*, which relates two planar surfaces via a linear transformation, in this case the road surface visible within camera frames and a suitable world coordinate system (We use Tennessee State Plane coordinates (EPSG:2274), which are preferred to other systems such as WSG84 (standard GPS convention) in that they utilize a globally orthonormal basis.) The road surface is treated as a planar surface (for each direction of travel) within a limited *field of view* (FOV) for each camera. Intrinsic-extrinsic camera calibration as used in the AV context [124, 284, 285] is infeasible here as cameras were not accessible prior to installation, can be replaced or moved, the focus is not fixed, and in-situ intrinsic camera calibration is not possible.

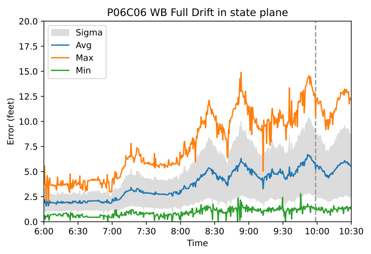
To compute each homography, lane marking corners are utilized as well-defined, semantically meaningful features. World coordinate system points are obtained by manually labeling aerial survey footage ($\sim$1 inch/pixel), while the corresponding image coordinates are produced with semi-automatic labeling on the recorded images. Manual aid was required because the lane markings are identical and repetitive, so additional visual clues were required to uniquely label each lane marking. The homography matrix is fit to these correspondence points via a least-squares formulation as implemented in OpenCV [348]. See Appendix O for details.

7.2.2.1 Homography Re-estimation

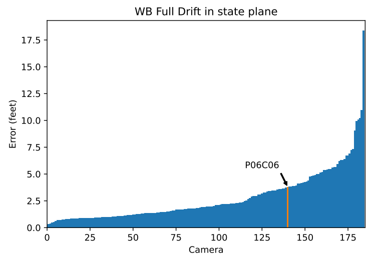
Ideally, homographies ensure that multiple views of the same point map to a single unique point on the state plane. In reality, camera fields-of-view are constantly changing due to inaccuracies in the pan-tilt mechanism during homing, settlement of the foundation, and most significantly the *sunflower effect* (the tilting of metal infrastructure poles away from the sun due to differential heating of the sun and shade-facing sides of the pole) [349]. Uncorrected, these factors produce significant homography errors sometimes greater than 10 feet. Figure 7.3a shows the magnitude of these shifts at one time for a typical camera homography. Figure 7.3b illustrates how the average shift for a camera changes over time, due to both the initial error (due to long term phenomena since the initial camera calibration) and the fluctuations in error over a single morning due to the rising morning temperature and changing cloud cover (peaks and valleys).



(a)



(b)



(c)

Figure 7.3: Typical homography error dynamic and the representation of the Sunflower Effect: **(a.)** uncorrected displacement of image points showing the magnitude of error (in feet) that using the original homography without accounting for drift would cause. The red polygon area represents the camera FOV, **(b.)** The displacement error of a typical camera over the day. Gray vertical line indicates the time instant shown in (a). **(c.)** Mean average displacement of all the cameras for the westbound roadway side, sorted by magnitude of error. (Credit: Gergely Zachár.)

Repeated manual correction is not feasible, and proper correction of the camera movement is challenging because traditional video stabilization methods (utilizing feature-matching techniques, based on e.g. SIFT [350] or SURF [351]) are ill suited for our scenes; a.) a large portion of the image corresponds to "noise" (e.g. trees, grass), producing hard-to-match feature points, b.) feature points are usually not semantically meaningful and potentially do not lie on the plane of the road surface, thus are unsuitable for homography estimation, c.) the relevant features on the ground plane in the region of interest are frequently occluded by vehicles, d.) a large number of co-moving vehicles can skew the calculation of optical flow along the direction of vehicle travel. To circumvent these issues, we propose the following homography re-estimation procedure:

1. Average frames for a suitable time ($\sim$ 1 min) to remove vehicles from the scene.
2. Find an initial, rough alignment based on a SIFT and a FLANN-based matcher [352] (as in OpenCV [348]).
3. Shift original correspondence points using rough alignment. Use to seed re-detection of lane markers.
4. Filter and refine the detected lane marker corner points.
5. Calculate the homography matrix using successfully re-identified corner points.

For a specific time instance this automatic re-detection and homography re-estimation often fails, either due to i.) lane marking occlusion in heavy traffic or ii.) failure of FLANN matcher. To provide a robust homography in spite of these failures, two methods are proposed and implemented: i.) calculation of a single, *static homography* for an extended period (e.g. all-day) by filtering and averaging homographies over the period, and ii.) a *dynamic, time-varying homography*. The later a time-varying kernel-based filter of the homography parameters, with a variable window size. Each method is computed offline (utilizing all information for the whole day). Additional details are given in Appendix P. The effectiveness of each solution is compared to the existing approach (FLANN-based matcher) in Section 7.3.1.

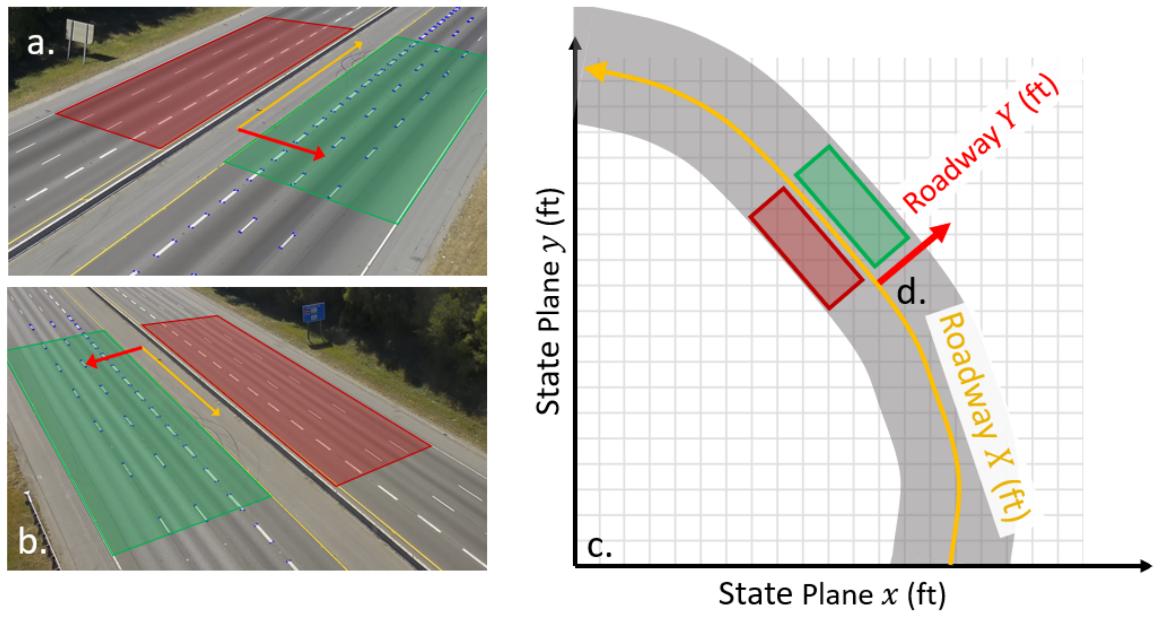


Figure 7.4: Camera fields of view (a and b) are related to (c) state plane coordinates, a rectilinear coordinate system, via perspective transforms. State plane coordinates are related to curvilinear roadway coordinates (d) via straightforward mathematical equations.

7.2.3 Roadway Coordinate System

We define an additional roadway coordinate system with the primary (X) axis aligned with the roadway direction of travel, and the secondary(Y) axis always perpendicular to the roadway direction of travel. Since the roadway is not perfectly straight, a *curvilinear coordinate system* is required to achieve the desired attributes, resulting in a locally orthonormal coordinate basis (see Figure 7.4 for a comparison). Such a coordinate system enables strongly domain-informed filter-based trackers [5] to be implemented trivially (e.g assume that the primary direction of motion for objects is along the primary axis and enforce reasonable vehicle physics). This coordinate formulation is also preferred for traffic science because quantities such as lane position and inter-vehicle spacing within a lane can be easily computed. A full description is given in Appendix O.

7.2.4 GPS Tracks and Correction

Concurrent with video recording, a fleet of 103 GPS instrumented vehicles was driven through the portion of roadway observed by the I-24 MOTION testbed. Details on vehicle instrumentation can be found in [13]. On these vehicles, positional data was recorded at 0.1s intervals. A total of 270 vehicle passes through the roadway were made during the recording period, providing the same number of vehicle trajectories for comparison.

7.2.4.1 GPS Track Refinement

Initial attempts to compare GPS track data against known, ground truth object positions (manually annotated) revealed that GPS data contained positional errors (mainly bias along primary direction of travel, and mainly high variance perpendicular to direction of travel), consistent with the GPS sensor's reported error metric of 2.5m *circular error probable* (CEP) (see Figure 7.5). Additionally, a small time discrepancy between some GPS track data and the camera network is observable. The following protocol was utilized to make GPS trajectories suitable for direct comparison against object tracking outputs from camera data:

1. Manually annotate a 'perfect' position for each GPS track, once per camera pole (e.g. 37+ annotations for a GPS track that travels the full length of the camera system). See Figure 7.5.
2. Correct GPS bias in the roadway coordinate system primary (longitudinal) axis direction by finding the mean offset between GPS positions and manually annotated object positions.

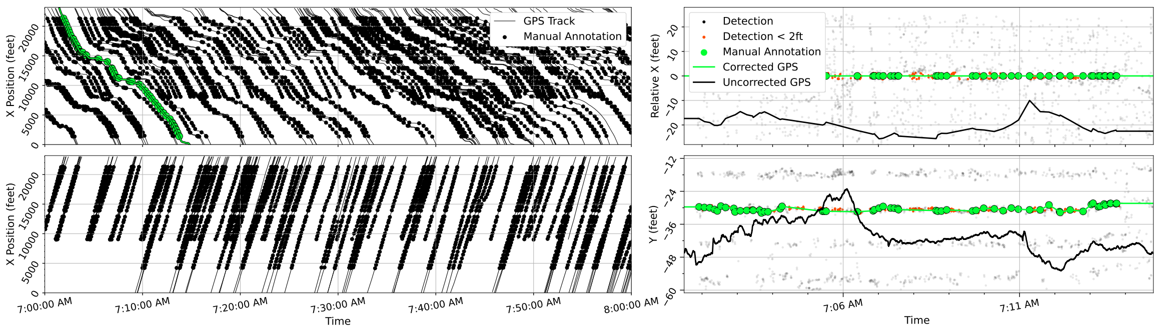


Figure 7.5: **(left)** GPS tracks (lines) and corresponding manual annotations (circles) for westbound (top) and eastbound (bottom) roadway directions of travel. One GPS trajectory is highlighted in green. **(right)** Detail for highlighted trajectory, showing relative x-position (top) and y-position (bottom) of nearby object detections (black dots), manual annotations (green circles), and the uncorrected corresponding GPS track. Deviations of over 20ft x / 12ft y position can be seen. Detections closely matching corrected GPS track shown in red. (Detections for every 30th frame are plotted for clarity.)

3. Determine the time offset in the range [-2s,2s] that minimizes the variance in GPS positional offsets relative to manually annotated object positions.
4. Correct residual error in the longitudinal direction by linearly interpolating the required offset between consecutively labeled offsets between GPS and manually annotated object positions.
5. Linearly interpolate lateral coordinate between manually annotated object positions for each GPS track.

Figure 7.5 shows the alignment between manually annotated object positions (circles) and GPS positions (lines) for a single typical GPS track. In total 7885 manual annotations are made. Figure 7.6 shows a histogram of GPS intersection-over-union alignment with object detections (see Section 7.2.5) before and after correction. Corrected GPS tracks align more closely with CNN-produced object detections than original GPS tracks (IOU of 45% vs 8%). After correction, 270 vehicle trajectories were produced with an average length of 6.6 minutes and 17560 feet. Each object is virtually always visible in at least one camera, corresponding to a minimum of 3207600 roughly annotated bounding boxes.

7.2.5 Detections

To allow preliminary analysis of existing object tracking methods, a baseline set of object detections was produced. Because the cameras in this dataset have widely varying fields of view, a viewpoint agnostic monocular object detector was utilized (i.e. an object detector that does not explicitly or implicitly code

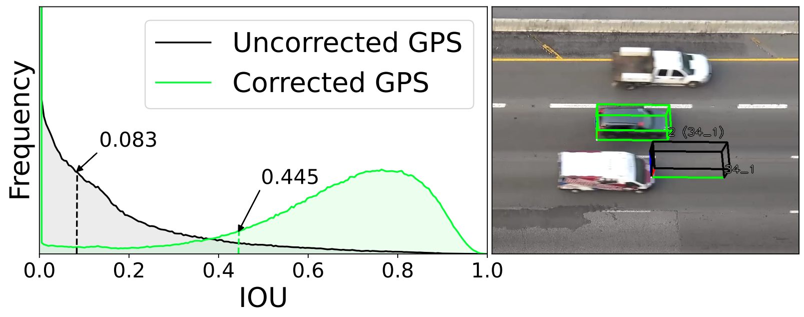


Figure 7.6: **(left)** Intersection-over-union histogram between GPS and closest automatically detected object position, before (black, mean 0.083) and after (green, mean 0.445) manual correction. **(right)** Examples of corrected (green) and uncorrected (black) GPS positions in a camera field of view.

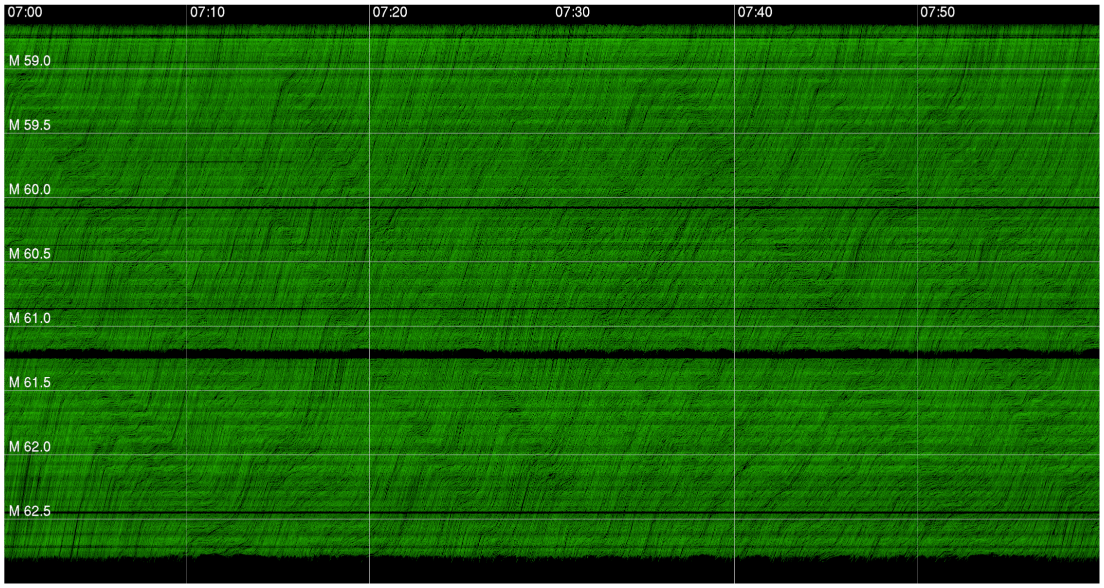


Figure 7.7: Longitudinal (X) position versus time for all detections on the westbound side. Each colored pixel (a total of 123,768,540 though with some overlaps for vehicles in different lanes but the same X position at the same time) represents a detected vehicle in a particular location and time.

scene information into its structure or parameter weights). This allows a single set of network parameters to be utilized for all camera fields of view (rather than training a separate model for each camera field of view, which was infeasible based on storage, implementation, and training time constraints). This work utilized a Retinanet ResNet50-FPN backbone object detector [132] to provide detections. The network outputs were parameterized to produce rectangular prism representations for 3D bounding boxes in addition to 2D bounding box outputs for predicted objects (see Chapter 5). Detections are nominally produced at 30 Hz with some frames skipped to provide $\pm$ 1/60s synchronization across all cameras. The resulting dataset contains 158,976,915 detections, each including a 3D bounding box defined in the roadway coordinate system, a 2D bounding box in image coordinates, vehicle class (*sedan*, *midsize*, *van*, *pickup*, *semi truck* or *other truck*), timestamp, camera, and detection confidence.

Figures 7.7 and 7.8 show the resulting detection set in roadway coordinates. Each diagram plots roadway X-position versus time for all detections on the given roadway direction. In some cases, more than one detection may be mapped to a single pixel because they correspond to two detections occupying nearly the same X-position at the same time, in different lateral (lane) positions.In these figures, horizontal bands without detections correspond either to missing camera poles (see Appendix U or overpasses.

7.3 Experiments

This section first describes experiments used to assess the accuracy of the homography re-estimation method proposed in this work, then describes initial MOT algorithm benchmarking performed using baseline object detections.

7.3.1 Homography Re-estimation

To assess the effectiveness of homography re-estimation methods proposed in Section 7.2.2.1, we utilize the homography goodness-of-fit (equation 7.1) which indicates how well the homography maps between the image plane and state plane, and the error metric defined in equation 7.2 which indicates average the positional error in points translated between the image plane and state plane via the computed homography. For each method, homographies are computed at 1 minute intervals overlapping by 50%. The computed homography's fitness is assessed according to:

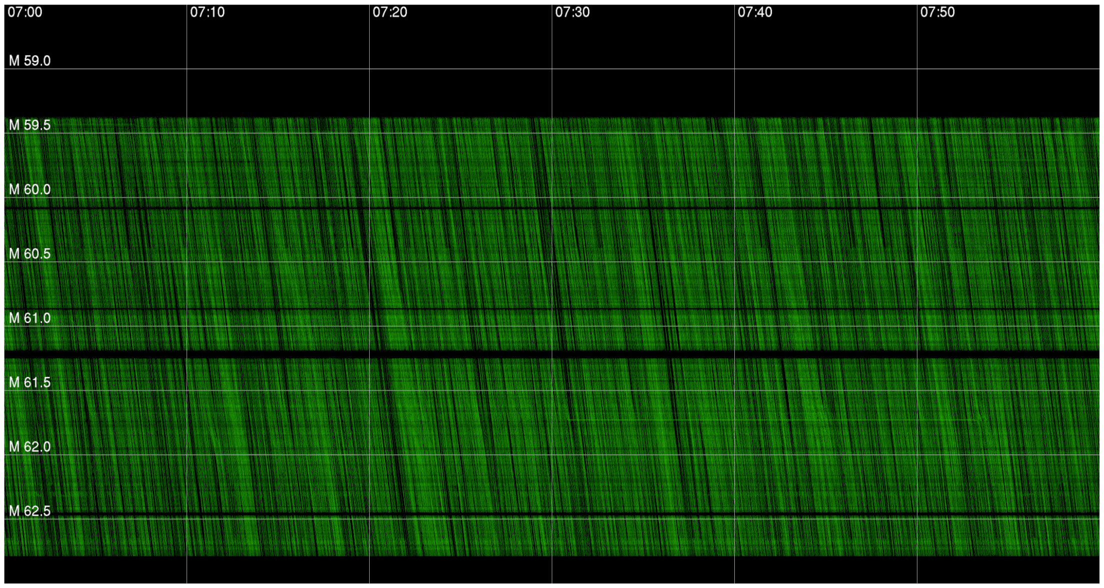


Figure 7.8: Longitudinal (X) position versus time for all detections on the eastbound side. Each colored pixel (a total of 123,768,540 though with some overlaps for vehicles in different lanes but the same X position at the same time) represents a detected vehicle in a particular location and time.

$$fitness(t) = ||\mathcal{A}_t, \mathcal{I}'_t \xrightarrow{H_t} ||_2 \tag{7.1}$$

where $\mathcal{I}'_t$ is the subset of correspondence points successfully rediscovered in the image at time t, A_t is the corresponding subset of points in state plane coordinates, $\xrightarrow{H_t}$ indicates a linear transform between coordinate spaces using H_t, the homography matrix fit directly to the rediscovered points at time t. Error is computed as:

$$error(t) = ||\mathcal{I} \xrightarrow{H_t}, \mathcal{I} \xrightarrow{H_t^*} ||_2 \tag{7.2}$$

where $\mathcal{I}$ is the full set of correspondence points labeled in the original reference image, H_t is the homography fit directly to time t between the rediscovered points I'_t and the corresponding state plane points A_t, and H_t^* is the homography for time t produced by the selected method. Because H_t is prone to error, any reported error may come either from the instantaneous homography H_t or the method-fit homography H_t^* (i.e. H_t is a good baseline when sufficiently many correspondence points are rediscovered.) We report other metrics independent from H_t in Appendix Q.

7.3.2 MOT Algorithm Benchmarking

A limited set of detection-fusion tracking algorithms (SORT [5], IOUT [166], KIOU [322], and ByteTrack with both Euclidean distance and IOU as similarity metric [226]) is implemented based on the criteria that i.) algorithms must not require retraining on the tracking data as no training data for object detection is made available, ii.) must not require additional inputs (e.g. appearance embeddings), and iii.) must be tracking by detection-based (not joint detection and tracking-based) methods. These criteria are necessary because, on a dataset of this size, generating auxiliary information or conducting one-off algorithm runs on all videos is prohibitively time-intensive. For comparison, an *oracle* tracker is implemented which selects all detections close to a GPS trajectory and linearly interpolates tracklet positions between these selected positions. The oracle represents performance theoretically obtainable using the existing set of object detections with a perfect motion model. This evaluation is merely a first step at gauging the difficulty of this dataset; we make annotations and evaluation protocols public so that researchers may evaluate their own algorithms and report state of the art performance.

Tracking methods are evaluated using recall, assigned IDs per ground truth trajectory, and *Multiple Object*

Tracking Precision (MOTP) in terms of both IOU and Euclidean distance from [216], *Longest Consecutive Subsequence* (LCSS) by distance and time from [353], and DetA, AssA, and HOTA from [218]. Because the dataset does not densely label objects, a false positive count cannot be obtained. Thus, the DetA metric from [218] is modified:

$$DetA^*_\alpha = \frac{TP}{TP+FN} \tag{7.3}$$

where TP represents the number of object positions that are matched to a ground truth position with at least α IOU overlap, and FN represents the number of ground truth object positions with no such match. We follow the rest of the protocol from [218] for calculating AssA and HOTA.

7.3.2.1 Evaluation Protocol

Each object tracker is run using the detection set from Section 7.2.5. GPS trajectories and detections from each camera are obtained at slightly different times. To account for this, tracking evaluation is performed at fixed 0.1 second intervals, and each GPS trajectory and object tracklet position is linearly interpolated at each evaluation time. Evaluation is performed as in [216]. For all metrics other than HOTA metrics, a lax IOU of 0.1 is required for an object tracklet and GPS trajectory to be matched.

7.4 Results

7.4.1 Homography Re-estimation Performance

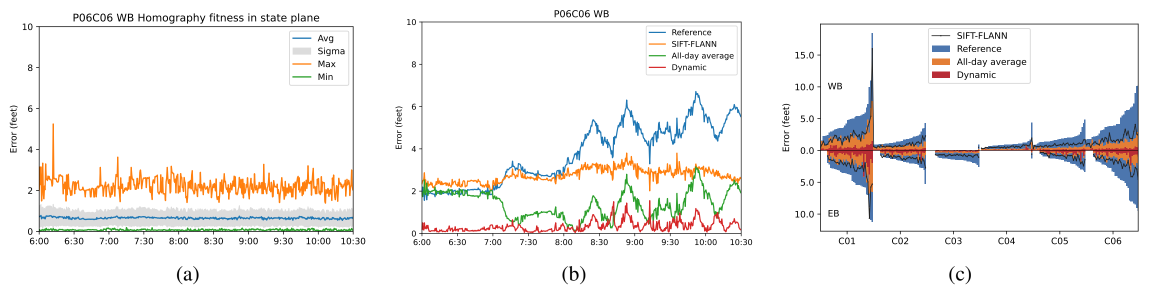


Figure 7.9: **(a.)** Typical homography *fitness* for a single camera, **(b.)** error dynamics for a single camera over time with each homography re-estimation methods, **(c.)** Remaining error for each camera after (black) SIFT-FLANN feature-matching, (orange) one-day best fit homography re-estimation, and (red) dynamic homography re-estimation methods relative to orignal reference homography baseline (blue). Cameras are grouped by position on pole (see Figure 7.2) and by side of roadway (westbound homographies on top, eastbound on bottom). (Credit: Gergely Zachár.)

Figure 7.9a reports the homography (H_t) goodness-of-fit metric (equation 7.1) over time for a typical camera using the homography re-estimation process defined in Section 7.2.2.1. This ∼2ft tightness is guaranteed by outlier removal processes during homography fitting; remaining error is due primarily to camera lens distortions and errors in the flat-plane assumption. The fitness of H_t represents an “error floor” for a homography based on the same assumptions.

Figure 7.9b show the additional error above the error floor for different homography re-estimation methods utilizing the error metric from equation 7.2. The reference (blue) indicates the resulting error without any mitigation, showing both long term (high mean) and short term (high variance) error (3.78 feet whole-day average). The SIFT-FLANN method (the existing optical flow-based ”camera stabilization” [348]), is inferior (2.74 feet whole-day average) in almost all cases to the proposed methods utilizing semantically meaningful lane markers. The static, all-day average homography removes the long term error, although it is mostly unable to remove the error caused by the *sunflower effect* especially in highly fluctuating cases (1.39 feet whole-day average). Lastly, the dynamic homography utilizes nearby (temporally) homography estimations for a given time instance, and can cope with short-term fluctuations caused by camera pole movement, substantially reducing (0.33 feet whole-day average) the residual error caused by the static homography.

Figure 7.9c compares the whole-day average error, per camera, for each homography re-estimation method. The SIFT-FLANN based method (black line) improves on the reference homography (baseline) for 98.6% of cameras. The static all-day reestimated homography (green) improves on the baseline for 100% of cameras and outperforms the SIFT-FLANN method for 88.1% of cameras. The dynamic homography method (red) improves upon the baseline in 100% of cases and on the SIFT-FLANN method in 99.7% of cases. The mean average error over all cameras is 2.78 feet for the reference homography , 1.42ft for the SIFT-FLANN method (49% reduction), 1.03ft for the all-day average method (63% reduction), and 0.33 for the dynamic method (88% reduction).

7.4.2 Multiple Object Tracking Performance

Tracker	**HOTA**	**DetA**	**AssA**	**Recall**	**IDs/GT** ↓	**LCSS**$_t$ (s)	**LCSS**$_d$ (ft)	**MOTP**$_i$	**MOTP**$_e$ (ft) ↓	**TD** (s)
SORT [5]	**9.5**	51.3	**1.8**	73.6	53.1	**51.9**	2609	68.0	**2.70**	12.3
IOU[166]	1.1	7.4	0.2	20.4	60.0	16.8	53.2	36.7	7.31	8.4
KIOU [166, 322]	8.5	51.2	1.4	73.9	**47.9**	40.6	2181	66.9	2.72	**15.1**
BT(L2) [226]	**9.5**	51.5	**1.8**	73.6	53.3	51.5	**2575**	**70.0**	2.71	12.4
BT (IOU) [226]	8.5	**53.1**	1.4	**75.9**	50.3	44.1	2390	67.1	2.72	14.9
Oracle	53.1	55.1	51.0	86.4	1.2	636	14699	75.3	2.53	690

Table 7.2: Tracking results for limited benchmark algorithm set. For each metric, a higher score is better unless indicated with a ↓. DetA and AssA indicate the detection and association components of HOTA, respectively. LCSS denotes the average longest consecutive subsequence (in seconds or feet) averaged across all trajectories. MOTP indicates the average precision (by IOU of object footprint or Euclidean distance) for all matched object bounding boxes, averaged over all trajectories. TD indicates mean tracklet duration.

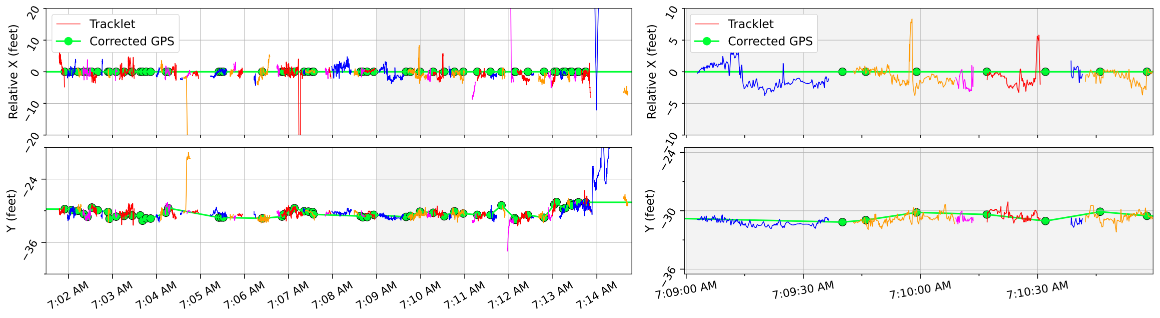


Figure 7.10: **(left)** A single trajectory (green) and all SORT [5] matched to this trajectory at least once (other colors). Manual annotations shown as green circles. **(right)** A close-up showing the LCSS matched to this trajectory (blue line), lasting ~32 seconds.

Table 7.2 shows multiple object tracking performance for the implemented trackers. First, note that HOTA is quite low for all trackers; driven primarily by low AssA scores. This indicates that object tracklets are not strongly persistent (this is also supported by the relatively low LCSS and mean tracklet durations compared to the 6.6 minute mean trajectory length, and high average IDs per ground truth). Such high fragmentation means the tracking outputs are not useful for traffic science applications requiring long and accurate object tracklets. All trackers with a motion model (all but IOU) achieve higher mean recall than raw object detections of 44.5% (see Figure 7.6), which indicates that the motion model is crucial for filling in object positional information when detections are missing. Figure 7.10 shows an example of all tracklets produced by SORT [5] matched to a single trajectory, and demonstrates the large number of tracklets associated with the ground truth trajectory.

The purpose of this initial benchmarking is not to claim that no existing tracker can perform well on the I24V dataset, but rather to show that popular off-the-shelf methods are not suitable without substantial enhancement such as more strongly physics and scene-informed models. For instance ByteTrack [226] achieves high performance on datasets such as MOTChallenge, where ID switches and fragmentations play a relatively smaller role in overall scores, but performs poorly on this dataset where object persistence plays a more outsized role in overall tracking performance, especially in the AsssA component of HOTA.

7.5 Conclusion

This work introduced the I24-Video Dataset, with concurrent video from 234 cameras recorded for one continuous hour capturing rush-hour traffic along 4.2 miles of interstate roadway, scene information for each camera, and 270 manually corrected GPS trajectories within the video data. These GPS trajectories were used to perform a preliminary benchmarking of object tracking algorithms, indicating that trackers utilizing stronger motion and appearance models are crucial for high performance on this dataset. The work also introduced new methods for keeping traffic camera homographies more precisely synchronized over time than existing methods allow. In the future, we plan to use this dataset to explore and design additional tracking algorithms that prioritize long term (10 minute, 18000 frame) object persistence, necessary for many traffic science applications. Several additional hours of GPS data are also recorded for future public benchmark competitions.

8. Conclusion and Future Perspectives

8.1 Conclusion

This dissertation proposed work to answer the following open question: *How can we create a large-scale traffic observation instrument for accurate and persistent vehicle trajectory generation using computer vision techniques?* Work to answer this question was proposed aligning with three major categories. First, a cutting-edge traffic instrument for rich data collection was proposed, designed, developed, and built. Second, computationally efficient algorithms for object detection and tracking across multiple cameras were developed. Third, several first-in-kind datasets were released, including a novel dataset enabling vehicle tracking in 3D space across a 17-camera network, a large-scale dataset featuring videos, scene homographies, detections and instrumented vehicle trajectories across 234 cameras, and the world's largest (temporally and spatially) vehicle trajectory dataset produced using the constructed instrument:

- **A Large-Scale Traffic Instrument for Trajectory Data Collection** This dissertation proposes, designs, tests and implements the *I-24 Mobility Technology Interstate Observation Network* (MOTION), a densely instrumented freeway that enables continuous, ongoing coverage of a roadway at the fine-grained vehicle trajectory level. MOTION consists of a network of 296 traffic pole-mounted 4K resolution cameras recording video data over a 4.2-mile stretch of freeway in its entirety. The raw video data stream exceeds 24 TB/day of traffic data footage that must be processed in real-time to extract precise vehicle locations, trajectories, and other relevant information from the entire monitored portion of roadway. This work discusses motivating design considerations, proposes a cost-effective physical sensor configuration to enable the data extraction requirements, presents preliminary experiments assessing the feasibility of the system, conducted as part of the first phase of the MOTION deployment, and provides a reference for the I-24 MOTION system as built.
- **Algorithms**
  - **Multiple Object Tracking.** A novel multiple object tracking method is proposed, utilizing the core intuition that predictable object motion yields strong object location priors *before* objects are detected at a given time. The proposed approach leverages this information to reduce the required computation to detect and track objects by cropping small image portions known to contain object priors and ignoring the rest of each frame. Experimental results show i.) the crop-based method increases both the speed and accuracy of an existing object tracker, ii.) the method achieves a new state of the art on the preeminent vehicle tracking benchmark [1], and iii.) the method yields a best result of 150% speedup with no decrease in accuracy.
  - **Vehicle Turning Movement Counting.** Based on the crop-based tracking method proposed, a novel intersection turning movement counting algorithm is proposed. Relative to the previous work, the primary contribution of this work is to introduce a new method for object initialization. Each manually identified source region is cropped at each frame and also processed by the same object detector as cropped objects, allowing for the detection of new objects without ever performing object detection on a whole frame. Experimentally, this method achieves competitive performance against other turning movement counting algorithms, and moreover increases speed by 57% relative to an otherwise identical method instead relying on full frame detections.
  - **Multiple Camera 3D Tracking.** The crop-based single camera tracking is extended to solve the multiple camera tracking problem. Relative to the previous work, the primary contribution of this work is to further reduce the image area that needs to be processed at each frame by leveraging the redundancy of overlapping camera views. Secondly, this work proposes a unified curvilinear coordinate system closely fit to the roadway lane markings capable of providing vehicle positional accuracy on the order of 2 feet across cameras while simultaneously aligning vehicle roadway motion along the primary coordinate system axis. Thirdly, this work proposes a new IOU-based loss formulation for 3D object detection from arbitrary viewpoints.
- **Datasets**
  - **Multiple Object Tracking Dataset.** This work proposes a new dataset that provides multiple object tracking labels in a shared 3D space, across multiple camera fields of view. Relative to

existing works, this dataset is the first to enable the development of 3D vehicle tracking methods in a traffic monitoring context. Moreover, the dataset has more synchronized camera views, more 3D vehicle bounding boxes, and more annotated frames than any other multiple camera multiple object tracking or traffic monitoring dataset. This work benchmarks a number of existing multi-camera multiple-object tracking approaches on the dataset, showing the implemented methods fall short of desirable performance especially in occluded and dense scenes.

– **Large Scale Vehicle Tracking Dataset.** This work proposes a new dataset designed to allow the benchmarking of object tracking algorithms for extremely long-term (10 minute) tracking performance. It consists of a single scene of video data, 1 hour in duration, simultaneously recorded from 234 overlapping cameras covering 4.2 miles of interstate roadway. A set 270 GPS trajectories recorded over 100 instrumented vehicles on the roadway during the recording duration is manually corrected to ensure positional accuracy. Initial experiments show that existing high-performing trackers fall well short of acceptable tracking performance on data of this scale, and further work is needed to develop suitable algorithms for long-term tracking tasks. Moreover, we take considerable care to make the data useful for computer vision applications, developing new techniques for keeping camera homographies more accurately aligned than existing stabilization methods allow.
– **Vehicle Trajectory Dataset.** 2 weeks of vehicle trajectory data are released with the original I-24 MOTION system paper [34]. Any given day of trajectory data is larger than all existing trajectory datasets (in terms of observation area length, duration of data recording and number of vehicles) and over time the I-24 MOTION trajectory data will be the only publicly available, continuous vehicle trajectory dataset except possibly among private entities.

8.2 Future Perspectives

8.2.1 Traffic Insights

The purpose of this dissertation work was to design and build the world's largest trajectory data instrument, but the larger goal of I-24 MOTION is to provide this data freely to enable new insights into traffic. I-24 MOTION generates trajectory data on a spatial and temporal scale that never available together before. This means that events that unfold over large spatial scales, but have fine-grained (vehicle dynamics level) causes or characteristics can be observed where they could not have been before. As an example, it may be possible to view a stopped vehicle or a collision in the trajectory data, and to trace the "wave-front" of the bottleneck backwards at the fidelity of a single car, observing also how the bottleneck spreads between lanes. Similarly, stop and go wave asynchronity across lanes can be observed (as noted in the preliminary analysis of Chapter 3, as well as wave splitting and merging phenomena, as well as anomalous vehicles that, while proportionally small in the overall flow of traffic such that they are lost in sensor modalities that aggregate data temporally, have an outsized effect on the local traffic dynamics around them). In short, we don't know what researchers may find, but it is easy to argue that the ability to observe a domain at new fidelity and scale has the potential to enable novel and disruptive findings.

8.2.2 Live Experiments

The instrument was also designed to support live experiments in traffic, including large deployments of automated vehicles which are designed to smooth traffic jams. The instrument will also support experiments conducted in collaboration with Tennessee Department of Transportation to support active traffic management, including experiments using variable speed limits, ramp meters, and lane closure systems. Such experiments will allow further investigation of the consequences of emerging technologies on traffic flow. As an example, the *Congestion Impacts Reduction via CAV-in-the-loop Lagrangian Energy Smoothing* (CIRCLES) experiments conducted in November 2022 aimed to utilize Level 2 connected and autonomous vehicles [354] to dampen stop and go waves on the corridor, reducing the overall energy consumption of vehicles in congested traffic [355]. Figure 8.1 shows images of instrumented and controlled vehicles during the experiment. The experiment relied on dense trajectory data produced by I-24 MOTION to analyze the energy consumption of every vehicle on the roadway during the test. Figure 8.2 shows gps-recorded data from this test superimposed on data from I-24 MOTION. Additional vehicle experiments on I-24 MOTION are slated for Winter 2024.

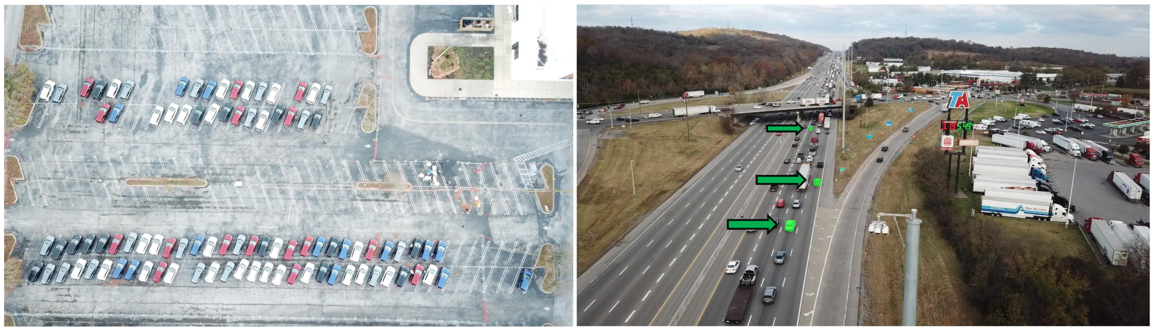


Figure 8.1: (left) The fleet of instrumented vehicles used to execute control strategies during the CIRCLES 2022 experiment. (right) Software-controlled instrumented vehicles (green) are visible on the roadway, and their positional data is recorded by I-24 MOTION cameras. (Drone imagery courtesy of Said ElSaid).

Additionally, the instrument is collocated with the I-24 Smart Corridor and a first-in-kind AI Decision Support System for variable speed limit control [356]. This proximity will allow for data from I-24 MOTION to be used both in the tuning of these AI algorithms (e.g. according to the wave propagation speeds observed on the corridor or the release rates for ramp metering deployments) as well as in the collection and analysis of data showing the effects of these technologies.

8.2.3 System Evolution

The algorithms proposed and implemented on the instrument in this work represent the first iteration of a living system to be iteratively improved over time. Because of the need for a functioning software system in November 2022 for the CIRCLES experiment, the original software stack for I-24 MOTION was designed and implemented on an accelerated timeline. The first video data from camera poles in the MOTION system was available just 2 months before the CIRCLES experiment, and data from the last camera poles was available just 1 week before the experiment. As a result, our team prioritized having a working software stack over having a mature software system free from problematic edge cases and with state-of-the art tracking performance.

Now, free from this time constraint, the I-24 MOTION software development team has more flexibility to re-imagine the original software for faster, more accurate, and more resilient performance. Already, considerable work has been aimed at correcting issues visible in the first version of I-24 MOTION data. The concerted work on homography error analysis and correction laid out in Chapter 7 was conducted in response to the severe camera misalignment visible in the original data (e.g. in Figure 3.21c); this work was able to reduce homography errors in 100% of camera fields of view. Figure 8.3 shows a histogram of object positions in xy-roadway coordinate space, both before and after this homography correction. Significant misalignment is visible in the original homographies especially at the regions between consecutive poles; such misalignments make it difficult to reliably track objects across the testbed. These issues have been largely addressed by the homography re-estimation work. Similarly, the original database architecture designed to store intermediate data between tracking and post-processing was discovered to be extremely computationally burdensome (i.e. conversion of the data between necessary formats takes 4 hours per day, which hampers efforts to make the data production pipeline run as close to real-time as possible).

In a similar vein, the following modifications to the first system implementation have been identified as promising directions for future research:

Offline Tracking Methods Early on in the design process for the first generation software stack for I-24 MOTION, a decision was made to use an online tracking algorithm motivated primarily by the long-term goal of a real-time performant system. Early experiments were based on SORT [5] and later iterations of the tracking algorithm were heavily based on IOU [166] and KIOU [322]. These algorithms require an explicit tradeoff between keep objects alive for long periods between detections (increasing the possibility of identity swaps especially if the motion model is not well-fit) and killing objects quickly when they are undetected for a few frames (increasing fragmentations and false negatives). Current I-24 MOTION data exhibits heavy

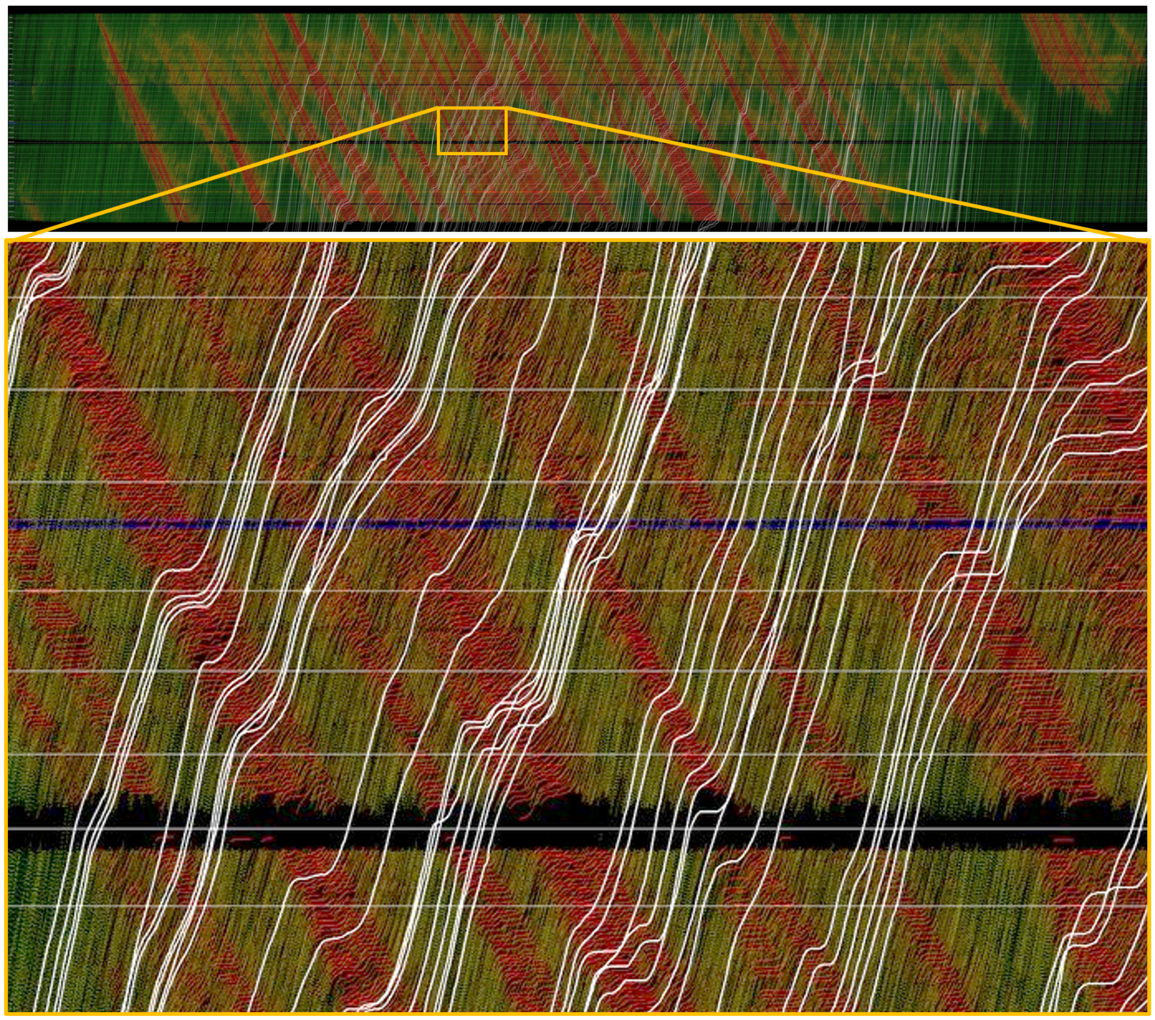


Figure 8.2: (top) Time-space diagram (y-axis is distance, 4.2 mi, and x-axis is time, 3 hours in total) showing data from instrumented vehicles (white) during the CIRCLES 2022 experiment superimposed over trajectory from I-24 MOTION. (bottom) An inset showing approximately 4000 feet and 20 minutes of data in more detail. Each vehicle trajectory can be traced through several stop-and-go waves during this period. (Credit: Gergely Zachár and Derek Gloudemans.)

object fragmentation, with mean object lengths on the order of [DG: Get] meters. Despite the popularity of filter-based online methods due to their simplicity and relative high-performance (especially in weak-physics domains such as image pixel-space), models that incorporate both past and future information *not only into the object positional estimation step but also into the object-to-detection matching step* wil likely outperform such filter-based methods (which require explicit detection-to-tracklet assignment prior to filtering). Offline tracking methods will be explored in future work.

While this decision does prevent the realtime running of this system, it is worth noting that i.) reasonably offline methods could be implemented in small temporal chunks, effectively approximating a fixed-lag realtime tracking system and ii.) the existing algorithms also run much slower than real-time currently but additionally can't leverage future information to improve tracking performance.

Combine Postprocessing and Tracking Similarly, the software architectural decision to postprocess data produced by the object detection and tracking step was borne of the poor performance object tracking. Future work will explore incorporating strong physics-based consistency into the offline tracking formulation across detections from all processing nodes, effectively eliminating the need for a separate postprocessing step.

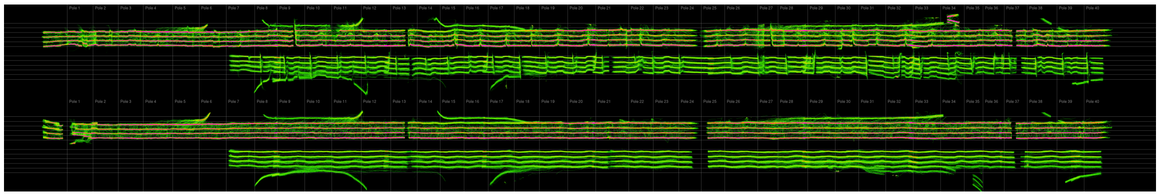


Figure 8.3: XY-roadway coordinate histogram for recorded vehicle positions during the I24-Video dataset recording duration (see Chapter 7). (top) before homography correction and (bottom) after homography correction. Best viewed in browser, zoomed in. (Credit: Gergely Zachár and Derek Gloudemans).

Faster Joint Object Detection Methods Modern object detectors are, for the most part, too slow for realtime performance on the I-24 MOTION system, as the system requires roughy 4 cameras to be processed on a single Nvidia A5000 GPU in realtime. One potential solution is to use simpler, less powerful, but faster object detectors and instead use the tracking domain as a strong prior to boost object detection performance or to alter object detector architecture. For instance, an object detector could be trained on the difference between consecutive frames, leveraging the fact that objects that enter a frame always result in a difference between consecutive frames in the region of appearance. In a related vein, inspired by [88, 207] multiple frames could be processed simultaneously with CNN architectures that enforce cross-frame object continuity.

Integrate Explicit Occlusion Models Relying on motion models for implicit occlusion handling has proven to be insufficient on this dataset (see Figure 3.21, where few object tracklets survive through 100-foot occlusion gaps). Explicit occlusion models as in [194, 209, 210] could be added to enforce object continuity (e.g. by modeling each occlusion zone as a rough queue and preventing object birth/death in these regions).

Extend Instrument Field of View onto Interchanges Additional cameras are allocated at interchanges to eventually allow for object tracking through interchanges as they enter and exit the interstate. Such a tracking task poses significant challenges, such as: i.) these portions of the roadway graded so their planes are changing and not parallel to the plane of the main interstate roadway, ii.) the coordinate system used throughout I-24 MOTION is defined relative to the yellow line for each roadway direction of travel; a new piece-wise curvilinear coordinate system would need to be defined to maintain reliable object positional information on ramps. Nevertheless, the inclusion of such data may reveal valuable traffic insights such as how ramp buildup can affect the flow of traffic on the roadway upstream.

Incorporate Appearance Information for Tracking Current I-24 MOTION tracking algorithms do not utilize appearance information to associate tracklets. This information represents low-hanging fruit by which to improve the performance of trajectory stitching during post-processing with minimal modifications to other portions of the pipeline. Joint detection and embedding models such as [170, 357] are well-suited towards this task.

8.2.4 Democratizing I-24 MOTION Data

I-24 MOTION data represents a unique challenge in that it is i.) of sufficient size that it requires some "big data" techniques to work with effectively, and ii.) is domain-specific and primarily of interest to traffic researchers who traditionally work with much smaller datasets. Over time, we hope to add the following data utilities to make the data more accessible and useful to researchers without having requiring a strong data-science toolkit.

For instance, visualizing entire days worth of trajectory data is not feasible with a scalable vector graphics library such as matplotlib. Rather, the data must be rasterized into a pixel-based image. Even then, many browsers and applications are limited in their ability to open extremely large images, meaning that the full resolution of the dataset must be downsampled for viewing convenience. For instance, in Chapter 3, we present downsampled versions of original images 30000 x 5940 pixels in size. Even at this larger scale, one pixel corresponds to a 2 foot / 0.5 second bin, so at high speeds trajectories can appear disconnected (visible in Figure 8.2. This issue is explored and preliminary applications are developed to circumvent these issues in [358], but in the future we would like to add additional functionalities to this tool.

Likewise, one day of trajectory data generally corresponds to >10GB of data, which is cumbersome to load into computer memory and which may not be feasible to open with traditional file management tools (e.g. Microsoft Excel). Our team plans to build a toolkit for working with this data using iterators so that only a small portion of the data needs to be kept in memory at any one time. This toolkit will also include additional tools for converting data batchwise between units and coordinate systems.

8.2.5 The Future of Vehicle Trajectory Collection

I-24 MOTION was conceived and implemented at an ideal time to leverage advances in computer vision algorithms and GPU-enabled computing techniques. We hope that I-24 MOTION represents a milestone in a golden era of traffic sensing fueled by the confluence of sensing and computing technologies. It is our hope that similar research efforts can benefit from the extensive work on designing infrastructure, hardware, and software systems for vehicle sensing, and we are excited by the efforts of projects such as ACTION, Zen Traffic Roadways, and the Lower Saxony Testbed [79, 80, 106] moving in the same technological current as us to provide a new generation of traffic sensing.

On the other hand, it is possible that such efforts may represent an inflection point in traffic sensing. The next generation of traffic researchers may never know the difficulty of limited vehicle trajectory data, and will have from the onset an extensive, longitudinal vehicle trajectory data repository and testbed on which to fit models, validate hypotheses, and deploy ITS solutions for testing. Perhaps future trajectory gathering efforts will focus more on collecting a more diverse cross-section of trajectory data in a greater variety of locations and conditions. In this vein, highD [66] is inspiring for the diversity of locations recorded, and pNEUMA [71] is exciting because it tackles the challenge of using a swarm of drones to vastly extend the field of view recorded relative to a single drone. We look forward to a day when drone-swarm based solutions are available to produce large-scale trajectory data at any location.

Regardless of the course traffic researchers chart in the next decade, this dissertation concludes with optimism for the insights, paradigm-shifting research, system improvements, and new ITS technologies that are all but guaranteed in the wake of the I-24 MOTION system as it stands today.

Figure 8.4: Cameras in operation.

Appendix

A List of Associated code repositories

- Code for LBT-Count AI City Challenge [92]
- Code for Crop-based Tracking [91]
- Code for I24-3D [95]
- Code for system deployment
- Code for Polygon IOU Loss [94]
- Code for roadway coordinate system from [95]

B I-24 MOTION Infrastructure Locations (Ch.3)

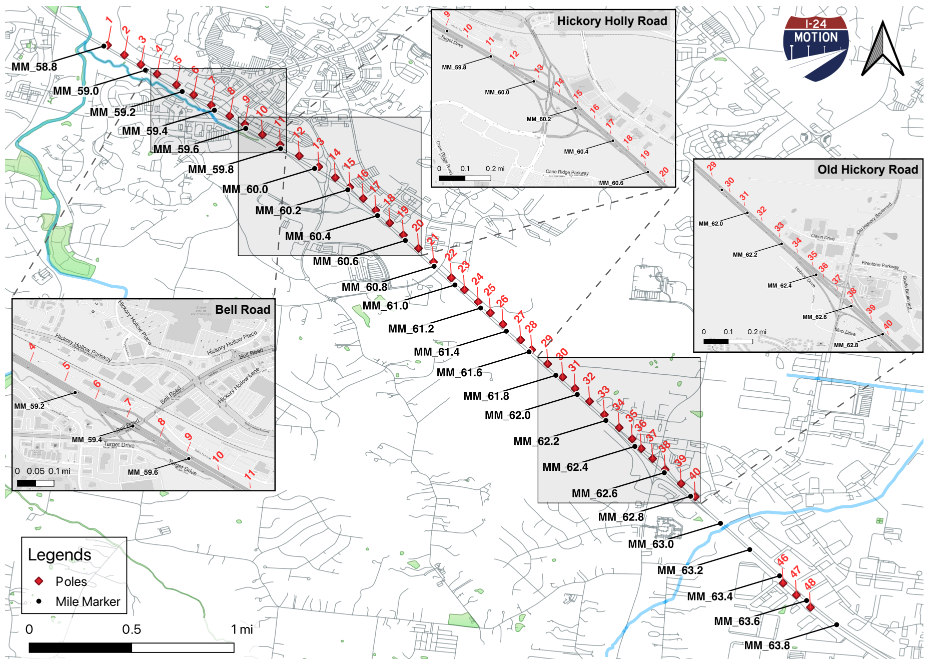


Figure B.1: Map for I-24 MOTION infrastructure locations

Appendix B is adapted from [34].

C Example Vehicle Trajectory (Ch.3)

Attribute	**Type**	**Unit**	**Value**
_id	12-byte BSON	–	63732b74e1fa5a45ae0c2fdd
vehicle class	int	–	0
first timestamp	float	s	1668436223.30
last timestamp	float	s	1668436257.60
timestamp	[float]	s	See Table 4
x position	[float]	ft	See Table 4
y position	[float]	ft	See Table 4
starting x	float	ft	325400.5531
ending x	float	ft	329300.5458
length	float	ft	15.6381
width	float	ft	5.8521
height	float	ft	4.7021
direction	int	–	1
Configuration ID	int	–	-1

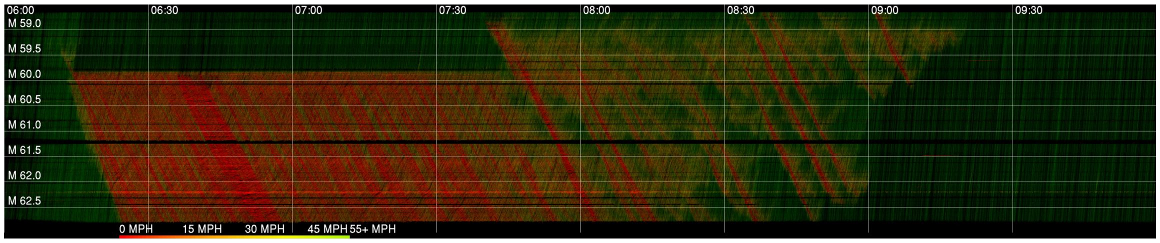
Table 3: Detailed information of the example trajectory.

timestamp (s)	**x position (ft)**	**y position (ft)**
1668436223.30	325400.5531	-19.19265508
1668436223.34	325405.0238	-19.12047988
1668436223.38	325409.4943	-19.04921183
1668436223.42	325413.9646	-18.97885093
1668436223.46	325418.4349	-18.90939717
...	...	...
1668436257.42	329281.8317	-43.03453987
1668436257.46	329286.5097	-43.09132499
1668436257.50	329291.1881	-43.14893520
1668436257.54	329295.8668	-43.20737050
1668436257.58	329300.5458	-43.26663087

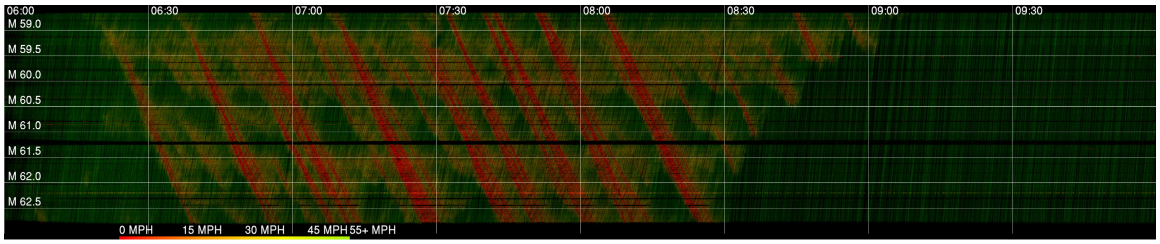
Table 4: The first 5 and the last 5 trajectory points for the example trajectory.

Appendix C is adapted from [34].

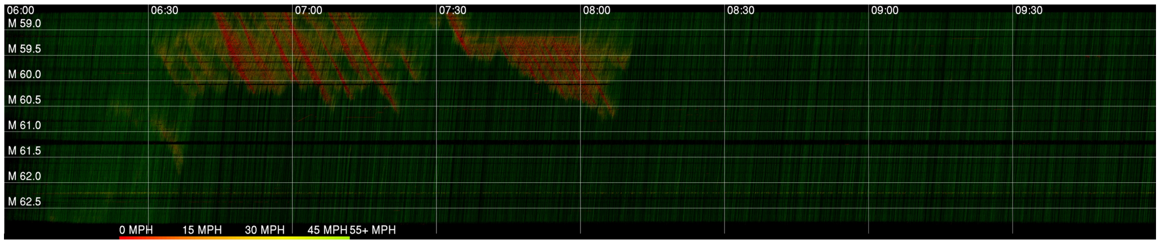
D Additional Time Space Diagrams (Ch. 3)



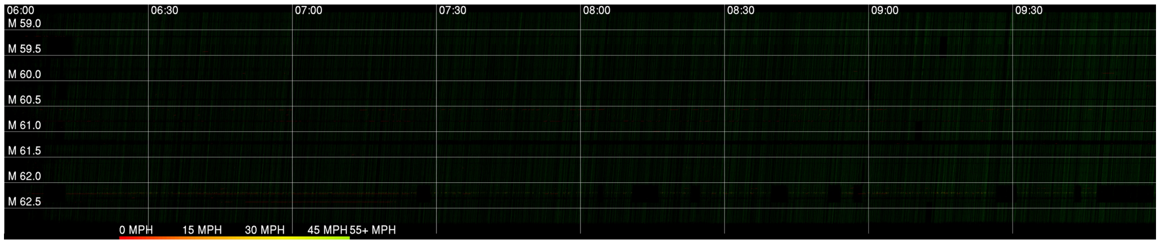
(a) Monday Nov 21 2022



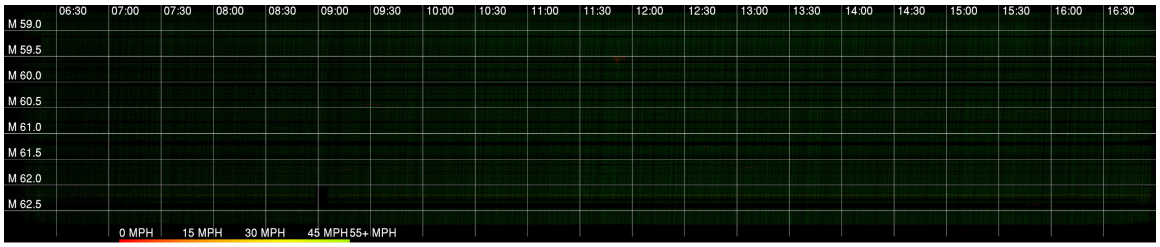
(b) Tuesday Nov 22 2022



(c) Wednesday Nov 23 2022



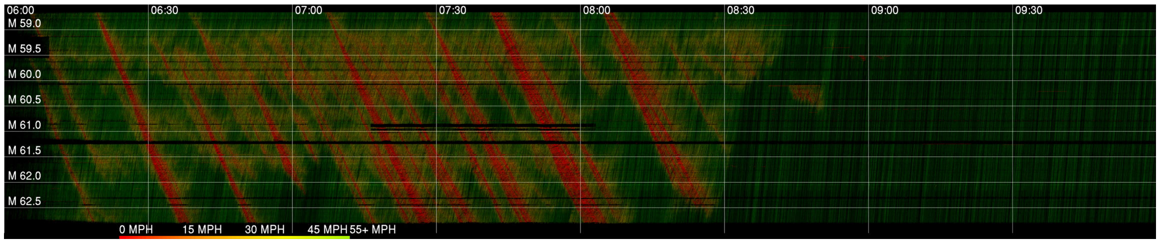
(d) Thursday Nov 24 2022 (Thanksgiving)



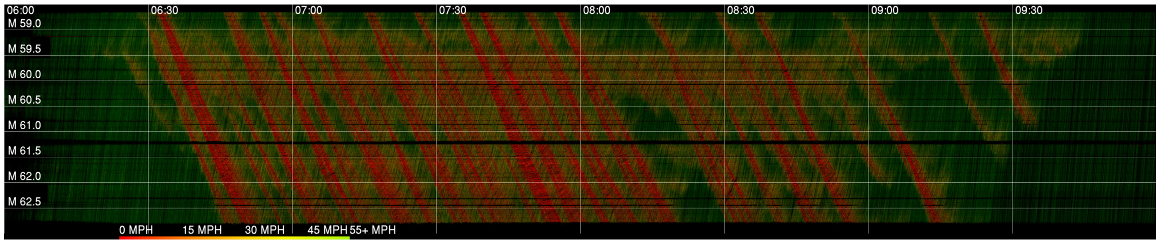
(e) Friday Nov 25 2022 (Black Friday)

Figure D.2

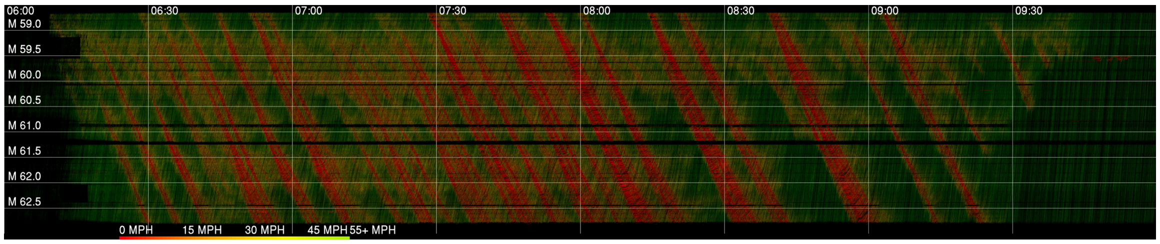
Appendix D is adapted from [34].



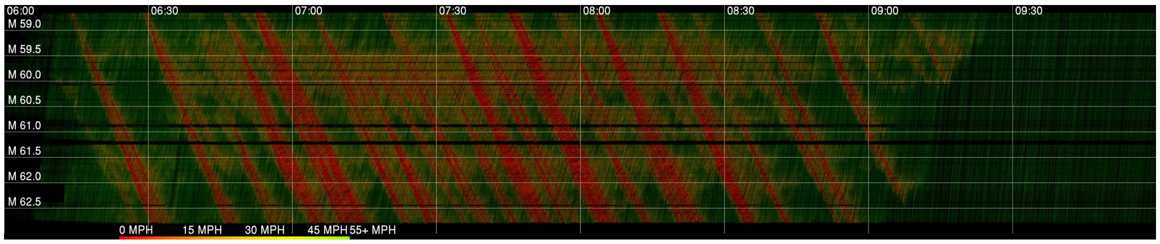
(f) Monday Nov 28 2022



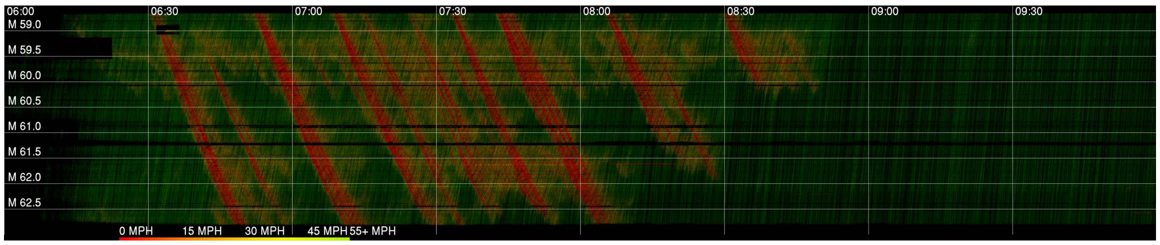
(g) Tuesday Nov 29 2022



(h) Wednesday Nov 30 2022



(i) Thursday Dec 1 2022



(j) Friday Dec 2 2022

Figure D.2: Additional time-space diagrams for I-24 westbound during morning rush hours on November (a)-(e) 21-25 and (f)-(j) November 28-December 2. (Credit: Gergely Zachár and Derek Gloudemans.)

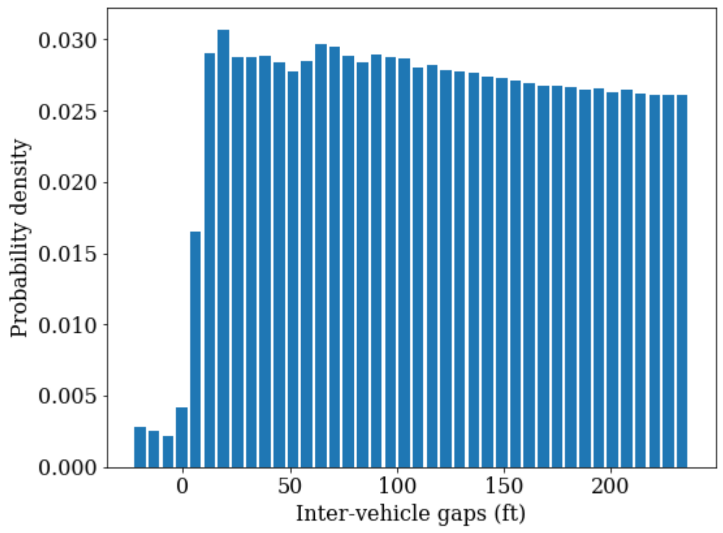


Figure E.3: Distribution of inter-vehicle gaps for westbound traffic during 6:00-10:00AM on Monday, November 21, 2022. Credit: Yanbing Wang.

E Platoon Consistency (Ch.3)

This work was primarily conducted by Yanbing Wang and is adapted from [34], and is included to show additional traffic analysis for which the I-24 MOTION trajectory data is suitable.

Platoon consistency refers to the physical consistency of inter-vehicle spacing resulting from the individual trajectories of two following vehicles [359].The leader-follower pair and their longitudinal gap are calculate at each timestamp. Figure E.3 shows a histogram of the inter-vehicle gaps distribution from the 4-hr westbound traffic data captured on November 21, 2022 as an example. Note that a small portion of the gaps are negative, and this is due to the artifacts mentioned above including a) homography error that causes multiple trajectories corresponding to the same vehicle, and 2) current data association step fails to connect partially overlapped trajectories of the same vehicle, creating the appearance of 2 vehicles when in fact there is a single vehicle. These artifacts cause overlaps in trajectories which may not accurately reflect the platoon consistency. Addressing these issues with more robust homography estimation and data association techniques is an on-going effort, and we expect many of these artifacts to be reduced in the next release.

F Traffic Wave Calculations (Ch. 3)

This work was primarily conducted by Yanbing Wang and is adapted from [34], and is included to show additional traffic analysis for which the I-24 MOTION trajectory data is suitable.

F.1 Wave propagation speed

The wave propagation speed is characterized by the slope of the slowdown that propagates upstream in the time-space diagram shown in 3.22. The slope is calculated based on the cross-correlation method as used in [314, 360], which compares the time series of the speed signals observed at two nearby locations on the same congested freeway. The idea is to shift one signal relative to another until the first non-trivial peaks are matched. The wave propagation speed is therefore the ratio between the time shifted and the distance of these two locations. We randomly select a few pairs of locations from one trajectory dataset and obtain a distribution of propagation speed. The distribution for the morning of Nov 22 2022, for example, has a mean of 12.8 mph and a standard deviation of 0.5 mph.

F.2 Wave frequency analysis

Wavelet transform is a time-frequency decomposition tool to effectively extract the non-stationary wave properties present in signals. The continuous wavelet transform is a convolution of the time-series signal $x(t)$ with a set of functions generated by the mother wavelet $\psi(t)$:

$$X_w(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{\infty} x(t)\psi\left(\frac{t-b}{a}\right) dt, \tag{1}$$

where $X_w(a,b)$ is a transformed signal at location b and scale a in the wavelet dimension. The scaling factor and the translation factor vary continuously, providing an over-complete representation of the signals. We select a commonly used mother wavelet as a Morlet wavelet:

$$\psi(t) = e^{-\frac{t^2}{2}} \cos(5t). \tag{2}$$

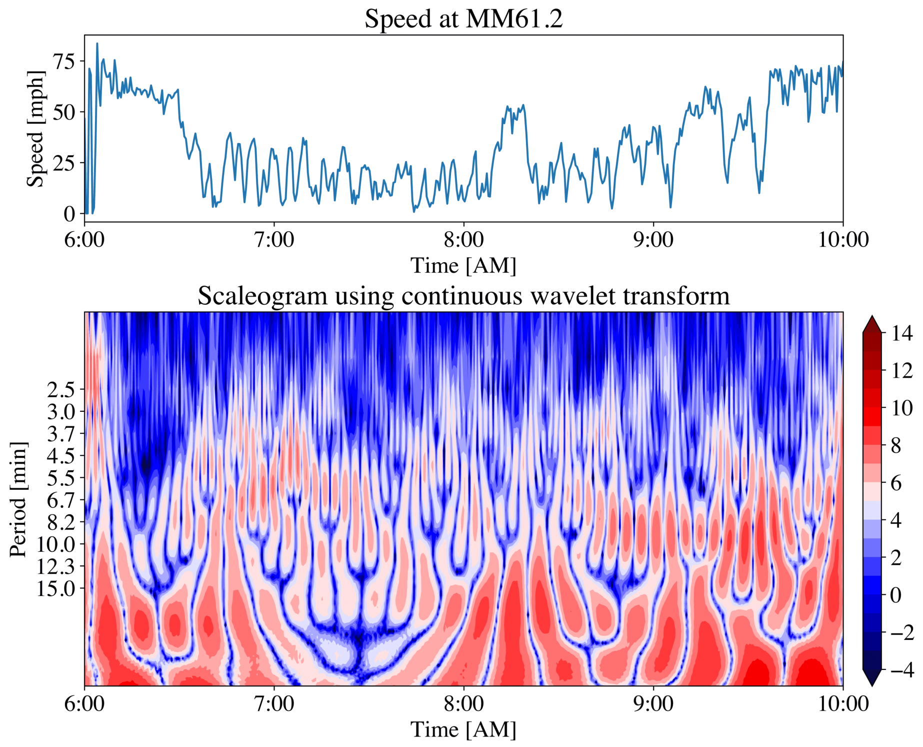


Figure F.4: Top: the speed time-series sampled from MM61.2 on Tuesday, Nov 29 2022. Bottom: a scaleogram produced by continuous wavelet transform of the speed signal. The color represents log-scale of the power distribution across both frequency and time domain of the signal. Credit: Yanbing Wang

An example of wavelet transform result is shown in Figure F.4. The top figure shows the time-series of speed sampled at a fixed location (in this case MM61.2) on Tuesday, Nov 29 2022. The bottom one is the corresponding wavelet transform scaleogram of the signal. It is obvious that the traffic waves do not appear to be stationary, i.e., the speed oscillation does not have a unique and consistent frequency across time. For example, during 6:50AM-7:30AM, the power of the signal peaks around 6.7min, corresponding to a salient wave period of 6.7min; during 8:30AM-9:30AM, the prominent wave period is near 9min.

G Kalman Filter Formulations (Ch.4 and Ch.5)

All standard Kalman filter notation in this Appendix is from [361] and [362]. Filter state, model and measurement matrices are fairly standard but are my own work.

G.1 2D Kalman Filter Formulation for Chapter 4
[1]

This section adopts standard filtering notation where the state of an object is expressed as $\mathbf{x}$. As with the Kalman filter described in [5], a constant velocity assumption (in terms of image pixel coordinates) is used to model object dynamics, and the state of an object at frame n, $\mathbf{x_n}$, is expressed as:

$$\mathbf{x}_n = [l_n, t_n, r_n, b_n, \dot{l}_n, \dot{t}_n, \dot{r}_n, \dot{b}_n]^T, \tag{3}$$

where (t,l) is the top left bounding box corner coordinate and (b,r) is the bottom right bounding box corner coordinate. The state variables $\dot{l}$, $\dot{r}$, $\dot{t}$, and $\dot{b}$ denote the rates of change for each respective side of the bounding box. (This state formulation is preferred to that described in [5] because empirically it results in improved object tracking performance.) The resulting system (written for a single object for simplicity) has the following state space form:

$$\mathbf{x}_{n+1} = \mathbf{F}\mathbf{x}_n + w_n\,, \quad \mathbf{y}_n = \mathbf{H}\mathbf{x}_n + v_n, \tag{4}$$

where $\mathbf{x}_n$ denotes the state at frame n, $w_n \sim \mathcal{N}(0, Q_n)$ is the process noise with covariance Q_n, $\mathbf{y}_n$ is the measurement at frame n and $v_n \sim \mathcal{N}(0, R_n)$ is the measurement noise with covariance R_n. The dynamical model $\mathbf{F}$ and the observation model $\mathbf{H}$ are written explicitly as:

$$\mathbf{F} = \begin{bmatrix} 1&0&0&0&\Delta&0&0&0 \\ 0&1&0&0&0&\Delta&0&0 \\ 0&0&1&0&0&0&\Delta&0 \\ 0&0&0&1&0&0&0&\Delta \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \end{bmatrix} \tag{5}$$

where Δ is the time step between consecutive video frames, assumed to be constant. Thus, the the model noise Q_n is determined for this fixed time-step by computing one-step error distributions associated with the model dynamics. The measurement noise H_n is computed separately for the crop detector and detector (to account for the differing qualities of bounding box accuracy) based on empirical measurement errors compared to true bounding boxes. The rest of the filter follows standard Kalman Filter formulation.

G.2 3D Kalman Filter Formulation for Chapter 5

This section adopts standard filtering notation where the state of an object is expressed as $\mathbf{x}$. In descriptive terms, a constant velocity along the roadway direction of travel is assumed, with assumed zero lateral velocity and zero heading angle. Each measurement consists of a vehicle (x,y) coordinate position on the roadway, and an estimate of each vehicle dimension. Each of these quantities is obtainable from 3D detector outputs as described in Chapter 5. The state of an object at frame n, $\mathbf{x_n}$, is expressed as:

$$\mathbf{x}_n = [x_n, y_n, l_n, w_n, h_n, v_n]^T, \tag{6}$$

where (x_n, y_n) is the back bottom rear center coordinate of the vehicle in roadway coordinate space, and (l_n, w_n, h_n) are the dimensions of the vehicle (length, width, and height in feet, respectively), and v_n is the object's velocity in the x-direction (in ft/sec). This state formulation is a modification of that described for SORT [5] adapted to 3D space. Note that the heading angle of the vehicle θ is assumed to be zero to maintain a linear and observable dynamical system. The resulting system (written for a single object for simplicity) has the same state space form as for the 2D case, with dynamical model $\mathbf{F}$ and measurement model $\mathbf{H}$:

[1] Appendix G.1 is adapted from [91].

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \Delta_i \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{7}$$

where Δ_i is the time between the last measurement update for object i and the current measurement for object i (which depends on the timestamp for the camera in which object i is measured). To use these 3D model dynamics in a multi-camera regime, the assumption that Δ_i is a constant is no longer valid. This is because the temporally closest frames from each camera are recorded at slightly different times (camera shutters are not synchronized). Thus, for each object i the *a priori* state must be computed at the specific timestamp for that camera frame. (Note that this introduces a slight peculiarity where we must select the best camera in which to query an object's location *before* we have predicted the updated *a priori* state for that object, which is done by using the object's last known position to evaluate equation 5.1. Thus, we need a method to express Let $\mathbf{Q}_i$ (the dynamical model covariance over this time difference Δ_i, which we cannot easily directly estimate) in terms of $\mathbf{Q}_n$ (model covariance over a fixed 1-frame timestep, which we can estimate with the error distribution of a series of time-step state roll-outs.

G.3 State Uncertainty $\mathbf{Q}_i$ with Nonuniform Time-steps Δ_i

This section is adapted from a Cross Validated Stack Exchange answer I authored in August 2022 [363]. Standard Wiener Process notation is used from [364].

The process (model) noise $\mathbf{Q}$ in a Kalman filter is assumed to be zero-mean Gaussian white noise. Under this assumption, the process noise at time t is independent from the process noise at $t + dt$. True fans of noise will recognize the time integral of Gaussian white noise as a Wiener process, or Brownian motion. There are some heavy-hitting (for me, but thankfully not for Albert Einstein [365] and Norbert Wiener [366]) mathematical derivations involved that conclude: *The covariance of a white Gaussian noise distribution scales with the square root of time.* How delightful to trail in some small way in the intellectual wake of such company!

Let $\mathbf{w}_n$ be process noise at time n, assumed to be drawn from a white noise (zero mean, time-uncorrelated) multivariate Gaussian distribution with covariance $\mathbf{Q}_n$.

$$\mathbf{w}_n \sim \mathcal{N}(0, \mathbf{Q}_n) \tag{8}$$

The state covariance $\mathbf{P}$ is increased during the prediction step according to:

$$\mathbf{P}_{n|n-1} = \mathbf{F}_n \mathbf{P}_{n-1|n-1} \mathbf{F}_n^T + \mathbf{Q}_n \tag{9}$$

$\mathbf{P}_{n|n-1}$ is the a priori state covariance estimate at timestep n given all observations up to $n-1$, $\mathbf{F}_n$ is the model dynamics at timestep n, and $\mathbf{P}_{n-1|n-1}$ is the *a posteriori* state covariance estimate at timestep $n-1$ given all observations up to timestep $n-1$.

Intuitively, the process noise covariance $\mathbf{Q}$ depends on time. Over an infinitesimal time, the process noise itself is infinitesimal, and likewise so is the process noise covariance. Likewise, over a large time period, the model noise must grow large (otherwise there would be no need for measurement!). Implicitly (in discrete time KF notation), $\mathbf{Q}_n$ represents the Wiener Process for an independent standard normal noise variable evaluated over one timestep. But we can also evaluate the Wiener Process over an arbitrary time window. Let $\mathbf{Q}(\Delta)$ represent the process noise covariance for an arbitrary time difference Δ, or the shift in the Wiener process over time.

A "timestep" is a discrete unit of time, but we require a continuous expression. To circumvent this, let Δ_n be the time difference between two consecutive timesteps, say n and $n-1$. When $\Delta = \Delta_n$, then $\mathbf{Q}(\Delta) = \mathbf{Q}_n$

**Let's move briefly into the notation adopted by Wiener Process [364].

The following expression holds for a Wiener Process W (here evaluated at times t_2 and t_1), where Z is an independent standard normal noise variable.

$$W(t_2) = W(t_1) * \sqrt{t_2 - t_1} \cdot Z \tag{10}$$

Rearranging, we obtain an expression for the change in a Wiener process over time:

$$W(t_2) - W(t_1) = \sqrt{t_2 - t_1} \cdot Z \tag{11}$$

Next, note that $\mathbf{Q}(\Delta)$ also represents the shift in a Wiener Process over time. We can thus write the expression for $\mathbf{Q}(\Delta)$ as:

$$\mathbf{Q}(\Delta) = \sqrt{\Delta} \cdot Z \tag{12}$$

which allows us to express the model covariance over an arbitrary time interval Δ in terms of the model covariance over the standard camera frame-rate (which is measurable).

$$\frac{\mathbf{Q}(\Delta)}{\mathbf{Q}(\Delta_k)} = \frac{\sqrt{\Delta} \cdot Z}{\sqrt{\Delta_k} \cdot Z} \tag{13}$$

finally yielding:

$$\mathbf{Q}(\Delta) = \sqrt{\Delta / \Delta_n} \cdot \mathbf{Q}(\Delta_n) \tag{14}$$

Thus, the model covariance over an arbitrary duration Δ scales with the (known) model covariance over a different (fixed) duration Δ_n according to the square root of the ratio of these two time durations. Intuitively, the integration of white process noise over time should scale less than linearly with respect to time because at each infinitesimal time-step the process noise is equally likely to be positive or negative (i.e. equally likely to add or subtract from the accumulated total noise). We would instead expect the scaling to approach linearity if the noise was highly time-correlated (i.e. a positive process noise at time t implies that the process noise at time $t + dt$ is likely also positive).

H Results on 2021 AI City Track 1 Challenge (Ch. 4)

H.1 AI City Challenge

Track 1 of the 2021 AI City Challenge requires multi-class, multi-movement vehicle counting on video sequences at intersections and along roadways. Thirty-one sequences from 20 distinct camera views are included, comprising about 9 hours of total video data all of which has resolution of at least 1280×960. Each camera field of view contains several vehicle movements of interest. To motivate the design of algorithms that can be evaluated in real-time on edge compute devices, the computational efficiency of vehicle counting algorithms are taken into account in addition to counting accuracy. Algorithms are assigned a score $S1$ according to the following formula:

$$S1 = 0.7 \times S1_{effectiveness} + 0.3 \times S1_{efficiency}$$

$S1_{effectiveness}$ uses cumulative vehicle counts at several times throughout each video sequence's overall length to evaluate counting effectiveness, weighting each time segment to help smooth jitters from vehicles counted near segment breakpoints. Cumulative count errors across all video sequences, turning movements and vehicle classes are normalized using the number of ground-truth vehicles within each cumulative count so that movements with more vehicles are counted more in the overall $S1_{effectiveness}$ score.

To partially account for the difference in operating speeds of various competitors' computing hardware, $S1_{efficiency}$ weights an algorithm's processing speed by the evaluating machine's speed at a set of benchmarking tasks relative to a baseline machine's speed on the same benchmark tasks. To compare these algorithms fairly in terms of speed, though, each algorithm must be run on the same compute hardware.

Appendix H is adapted from [92].

One half of the testing data is made available to challenge participants, with only a very small proportion of ground-truth labels provided such that supervised learning methods cannot feasibly be used. All submitted algorithms are evaluated on the full dataset, run on the same edge device (Nvidia Jetson NX development kit board.) As of submission, only aggregate $S1$ metrics from the first 50% of testing data are made public, so we report these scores in Table 5.

H.2 Parameter Settings and implementation Details

We use a Pytorch implementation of Retinanet with a ResNet50-FPN backbone for feature extraction. Because the scale of objects in image crops varies significantly from the scale of objects when detecting on whole frames, our localizer only is retrained for truck and car bounding box and class prediction per guidance from challenge organizers. Training makes no use of AI City Challenge data in any way. All code is run on a single GPU and 2 CPU cores (one of which is exclusively used for video decoding and frame buffering).

H.3 Track 1 Leaderboard

Team ID	Rank	S1 Score
37	1	0.9467
5	2	0.9459
8	3	0.9263
19	4	0.9249
118	5	0.9235
42	6	0.9157
95	**7**	**0.8576**
134	8	0.8449
153	9	0.8205
168	10	0.7545
144	11	0.7521
64	12	0.7506
86	13	0.6677
131	14	0.6548
133	15	0.4804
48	16	0.4205
77	17	0.3757

Table 5: $S1$ score for algorithms on 50% of testing data, evaluated on disparate machines.

Table 5 reports a comparison of all 17 algorithms submitted to the public Track 1 Challenge as part of the 2021 Nvidia AI City Challenge. Our algorithm (Team ID 95) places 7th in terms of $S1$ score on the 50% of testing data made publicly available, with $S1 = 0.8576$, $S1_{effectiveness} = 0.8549$ and $S1_{efficiency} = 0.8637$. LBT-Count processes the available videos at an average of 72.6 frames per second on a single GPU and 2 CPU cores.

H.4 Speed Comparison to Tracking by Detection

Lastly, to benchmark the impact of using Localization-based Tracking (LBT) rather than tracking-by-detection (TBD) for the object detection and tracking portions of our counting method, we implement a detect-track-count algorithm based on KIOU object tracking [166, 322]. We measure the speed of each method when a measurement step is performed at every frame. The same network structure is used for the localizer in LBT and the detector in TBD (Retinanet with ResNet50-FPN backbone). Table 6 reports the results.

LBT-Count is 52% faster than the detect-track-count (TBD) approach overall (20 fps vs 13.2 fps average). LBT is faster than TBD on 29 of 31 available test sequences, and achieves at least a 100% speedup on 19 of 31 sequences. The speedup of LBT is somewhat correlated to the number of crops (the sum of the number of tracked objects and the number of source regions for a camera field of view), as each cropped region

Sequence	Speedup	Crops	LBT-Count fps	TBD fps
cam_14	**534%**	2.3	**30.9**	4.9
cam_16	**308%**	2.5	**40.8**	10.0
cam_17	**310%**	2.8	**40.5**	9.9
cam_20	**308%**	3.5	**39.3**	9.6
cam_19	**309%**	3.6	**38.3**	9.4
cam_18	**323%**	3.7	**39.1**	9.2
cam_13	**252%**	4.9	**37.7**	10.7
cam_15	**283%**	5.4	**36.8**	9.6
cam_1_dawn	**171%**	5.6	**42.0**	15.5
cam_12	**251%**	5.9	**36.5**	10.4
cam_2_rain	**97%**	6.0	**39.0**	19.8
cam_10	**251%**	6.4	**37.0**	10.5
cam_1_rain	**166%**	6.5	**40.7**	15.3
cam_1	**150%**	6.6	**38.2**	15.3
cam_2	**64%**	7.0	**29.7**	18.1
cam_3_rain	**94%**	7.8	**37.1**	19.1
cam_3	**42%**	7.9	**25.4**	17.9
cam_11	**249%**	9.2	**35.8**	10.3
cam_9	**223%**	9.4	**33.7**	10.4
cam_8	**231%**	12.0	**34.2**	10.3
cam_7_dawn	**131%**	15.6	**33.9**	14.7
cam_4_rain	**102%**	16.5	**30.7**	15.2
cam_6_snow	**158%**	18.6	**32.9**	12.7
cam_4_dawn	**68%**	19.1	**25.7**	15.3
cam_4	-13%	19.5	12.7	**14.6**
cam_6	**35%**	21.5	**18.3**	13.6
cam_5_dawn	**78%**	21.5	**24.8**	13.9
cam_7_rain	**89%**	21.6	**26.1**	13.9
cam_5_rain	**81%**	24.6	**25.2**	13.9
cam_7	**9%**	27.3	**12.0**	11.1
cam_5	-12%	27.9	11.9	**13.5**
Average	**52%**	11.4	**20.0**	13.2

Table 6: Speedup from using LBT-Count versus a tracking-by-detection-based counter (TBD). "Crops" indicates the average number of cropped regions processed by the localizer per frame in LBT-Count.

requires additional computation to localize vehicles within it. Sequences with fewer than 19 crops per frame on average exclusively experience an increase in speed as a result of using the LBT framework.

I Scene Homography for I-24 3D Dataset (Ch. 6)

I.1 3D Perspective Transform Fitting

A *homography* relates two views of a planar surface. For each camera in each scene, we provide homography information such that the 8-corner coordinates of the stored 3D bounding-box annotation can be projected into any camera view for which the vehicle is visible, creating a monocular 3D bounding box within that camera field of view. For each direction of travel in each camera view, for each scene, a homography relating the image pixel coordinates to the roadway coordinate system is defined. (Though the same cameras are used for different scenes, the positions of the cameras changes slightly over time due to pole expansion and contraction in the sun). A local flat plane assumption is used (the roadway is assumed to be piece-wise flat) [344]. A series of correspondence points series of correspondence points $p_q = [x, y, x', y', z']$ are used to define this relation, where (x, y) is the coordinate of selected correspondence point q in pixel coordinates (row, column) and (x', y', z') is the selected correspondence point in roadway coordinates.

Appendix I is adapted from [95].

All selected points are assumed to lie on the road plane, so $z' = 0$ for all selected correspondence points. Visible lane marking lines are used as correspondence points in each camera field of view. Each lane is reliably known to be 12 feet wide, and each lane-separating tick mark is known to be 10 feet long and at a regular spacing of 40 feet. Thus, the road plane coordinates of each lane tick mark are known precisely. The corresponding pixel coordinates are manually selected in each camera field of view, for each direction of travel on the roadway.

A *perspective transform* (Equation 16) is fit to these correspondence points. We first define a 2D perspective transform which defines a linear mapping (Equation 15) of points from one plane to another that preserves straight lines. The correspondence points are then used to solve for the best perspective transform $\mathcal{H}$ as defined in equation 16, where s_i is a scale factor.

$$s_q \begin{bmatrix} x'_q \\ y'_q \\ 1 \end{bmatrix} \sim \mathcal{H} \begin{bmatrix} x_q \\ y_q \\ 1 \end{bmatrix} \tag{15}$$

where $\mathcal{H}$ is a 3×3 matrix of parameters:

$$\mathcal{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{16}$$

For each camera field of view and each direction of travel, the best perspective transform $\mathcal{H}$ is determined by minimizing the sum of squared re-projection errors according to equation 17 as implemented in OpenCV's $find\_homography()$ function [348]:

$$\min_{\mathcal{H}} \sum_q \left(x'_q - \frac{h_{11}x_q + h_{12}y_q + h_{13}}{h_{31}x_q + h_{32}y_q + h_{33}} \right)^2 + \left(y'_q - \frac{h_{21}x_q + h_{22}y_q + h_{23}}{h_{31}x_q + h_{32}y_q + h_{33}} \right)^2 \tag{17}$$

The resulting matrix $\mathcal{H}$ allows any point lying on the plane within the camera field of view to be converted into roadway coordinates and, the corresponding matrix $\mathcal{H}_{inv}$ can easily be obtained to convert roadway coordinates on the plane into image coordinates. However, since each vehicle is represented by a 3D bounding box, the top corner coordinates of the box do not lie on the ground plane. A 3D perspective transform $\mathcal{P}$ is needed to linearly map coordinates from 3D roadway space to 2D image coordinates, where $\mathcal{P}$ is a 3×4 matrix of parameters:

$$\mathcal{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tag{18}$$

and $\mathcal{P}$ projects a point in 3D space (x', y', z') into the corresponding image point (x, y) according to:

$$\mathcal{P} \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \sim s' \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{19}$$

By observing the case where $z' = 0$, it is evident columns 1,2, and 4 of $\mathcal{P}$ are equivalent to the columns of $\mathcal{H}_{inv}$ and can be fit in the same way. Thus, we need only solve for column 3 of $\mathcal{P}$. Next, we note as in [344] that $(\frac{p_{11}}{p_{31}}, \frac{p_{21}}{p_{31}})$ is the vanishing point (in image coordinates) of perspective lines drawn in the same direction as the roadway coordinate x-axis. The same is true for the 2nd column and the roadway coordinate y-axis, the 3rd column and the roadway coordinate z-axis, and the 4th column and the roadway coordinate origin.

Thus, to fully determine $\mathcal{P}$ it is sufficient to locate the vanishing point of the z-axis in roadway coordinates and to estimate the scaling parameter p_{33}. The vanishing point is located in image coordinates by finding the intersection point between lines drawn in the z-direction. Such lines are obtained by manually annotating vertical lines in each camera field of view. The scale parameter is estimated by minimizing the sum of squared re-projection errors defined in equation 20 for a sufficiently large set of roadway coordinates and corresponding, manually annotated coordinates in image space.

$$\min_{p_{33}} \sum_q \left(x_q - \frac{p_{11}x'_q + p_{12}y'_q + p_{13}z'_q + p_{14}}{p_{31}x'_q + p_{32}y'_q + p_{33}z'_q + p_{34}} \right)^2 + \left(y_q - \frac{p_{21}x'_q + p_{22}y'_q + p_{23}z'_q + h_{24}}{p_{31}x'_q + p_{32}y'_q + p_{33}z'_q + h_{34}} \right)^2 \quad (20)$$

The resulting 3D perspective transform $\mathcal{P}$ allows for the lossless conversion of points in roadway coordinates to the corresponding points in image coordinates.

I.1.1 Curvature Correction

Lastly, we fit a 2nd-order polynomial curve $f(x')$ to describe the y-coordinate along a solid lane line as a function of the x-coordinate (in space). We then use the fit curve to shift the y-coordinate of each point when it is transformed from image space to roadway coordinate space to account for lateral roadway curvature according to Equation 21.

$$y'_{offset} = y' - f(x') \quad (21)$$

where y'_{offset} is the corrected y-coordinate and $f(x')$ is the fit 2nd-order polynomial. We analogously perform the reverse correction when converting points from 3D roadway coordinate space into 2D image space. We note that the curvature also results in a slight error in x-coordinates, but the component of error in the x-direction is negligible for the purposes of this dataset as it is proportional to the sine of a relatively small angle (angle of roadway turn within the camera field of view). Figure I.5 shows coordinate system alignment to the roadway markings before and after curvature correction.

J Sources of Timestamp Error and Corrections for I-24 3D Dataset (Ch. 5)

We discover, through numerous tests, that the timestamps reported by the IP camera firmware, are inaccurate, approximately on the order of 0.1-1s. This Appendix describes the types of error we identify, then details the corrections used to partially compensate for these errors. Lastly, it presents error metrics before and after these timestamp corrections.

Appendix J is adapted from [95].

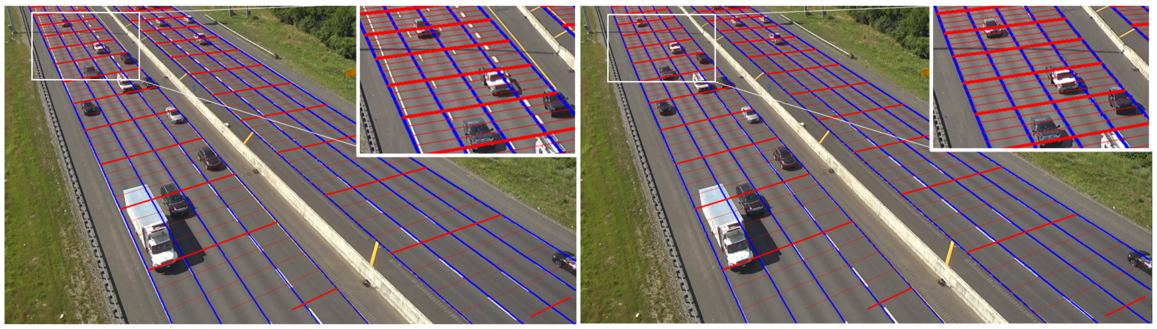


Figure I.5: (Left) Before curvature correction, lines in the x-direction (blue) with equal y deviate significantly from roadway direction (white and yellow painted lines on roadway) at distances far from points used to fit local camera homography. (Right) After curvature correction, roadway lines in image have equal y-coordinate.

J.1 Known Timestamp Errors

Figure J.6 provides a visual overview of 4 timing errors (camera phase differences, camera clock offset, camera timestamp quantization, and doubled/skipped frames) in the I24-3D dataset. We then provide an example of each.

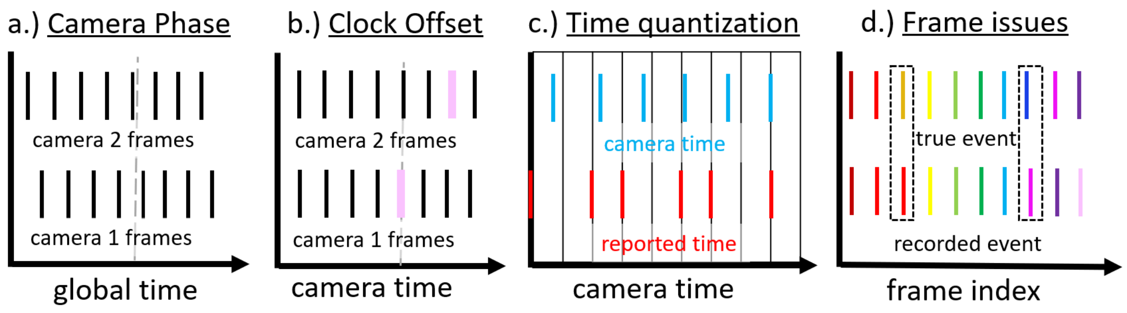


Figure J.6: Multiple-camera timing issues. a.) Frames from different cameras are out of phase. b.) Different cameras report the same event (pink) as occurring at different times. c.) Camera time (blue) is quantized to a lower precision (red) before being reported. d.) Cameras report the same frame twice or skip frames (dash box).

Camera Phase and Offset: Figure J.7 shows the closest (in time) frames from camera p1c5 and p1c6; the reported timestamps for these frames are still 0.01s apart, indicating the cameras record frames out of phase. Furthermore, though camera p1c5 reports a time 0.01s earlier than p1c6, the position of vehicles suggest that this frame was recorded after the frame from p1c6 (vehicles are slightly further along the roadway in their respective directions of travel). We suspect this is due to camera clock offset (bias) relative to one another.

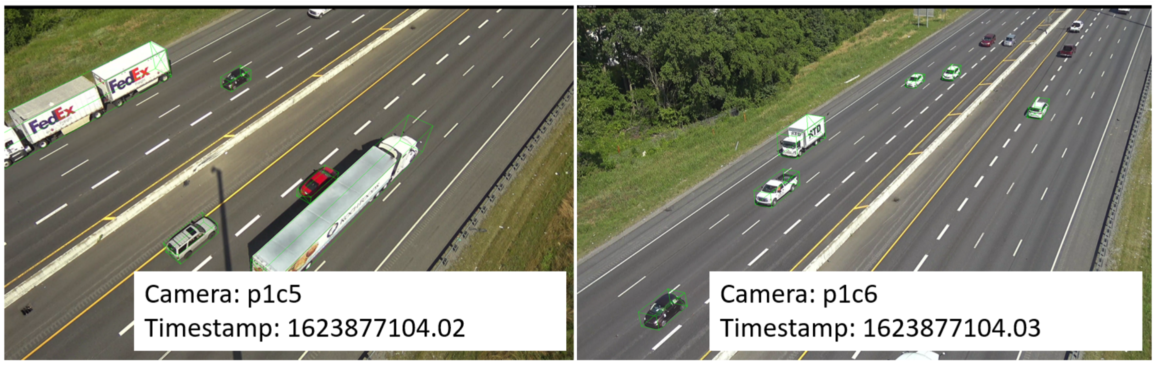


Figure J.7: Example of phase and clock offset error. Closest reported timestamps from cameras are unequal (phase), and the positions of vehicles in camera p1c5 suggest this frame was recorded later than p1c5 (clock offset).

Camera Timestamp Quantization Camera timestamps are reported to 0.01s precision, with higher-precision camera times quantized (rounded or truncated) before reporting. To demonstrate this behavior, we conduct the following test. We create two rotor clocks with two rotating disks (rotating at 2 Hz and 5 Hz). We control the rotational frequency of each rotor precisely using a dynamometer. We record the rotor clock rotation using a camera of the same model as deployed on I-24 MOTION (used to obtain the video data in this dataset). Based on the rotational position of each hand extracted using image processing techniques, we precisely determine the time *deltas* (Δ) (difference in time between consecutive frames). We compare the time deltas obtained by the coarse (2Hz) and fine (5Hz) rotor clocks to the reported timestamps from the camera.

Figure J.8 shows the clock setup with rotor positions, and Figure J.9 shows the reported frame deltas. The true time at which frames are recorded can be estimated from the coarse and fine rotor clock times, which adhere closely to the 30 fps nominal framerate (subject to some jitter due either to slight camera deviations from the nominal framerate or slight errors in rotor positional extraction algorithm). Conversely, the reported clock quantized timestamp deltas fluctuate significantly around the true clock times. Doubled frames are visible (when delta = 0s, the same frame and timestamp has been sent twice by the camera). Thus, the quantization of timestamps is shown to create error in determining the true time at which a frame was recorded.

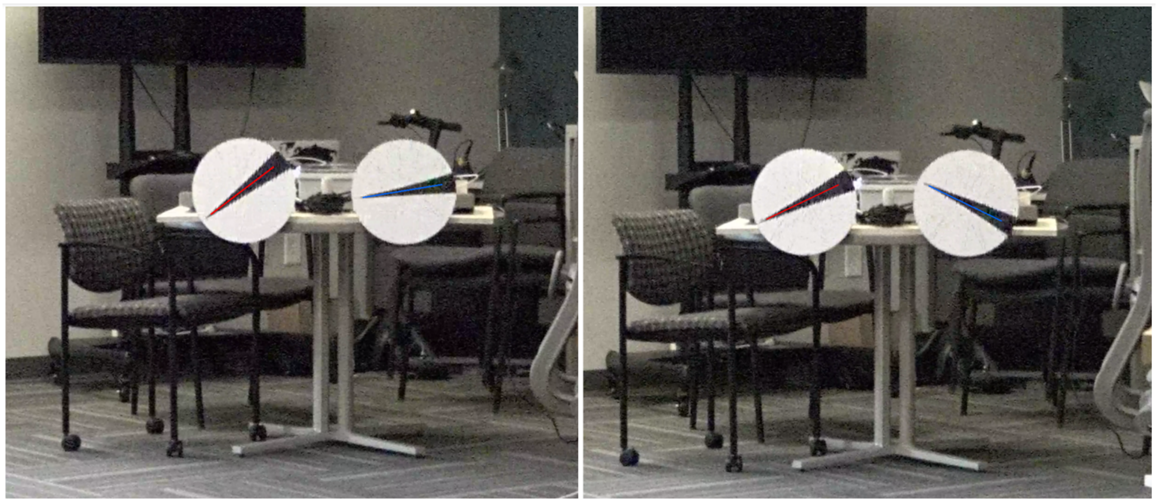


Figure J.8: Coarse 2Hz (red) and fine 5Hz (blue) rotors shown for two frames of lab test video. The position of each rotor along with a total rotor revolution count can be used to determine the time since the start of the video sequence and the time delta between two consecutive frames.

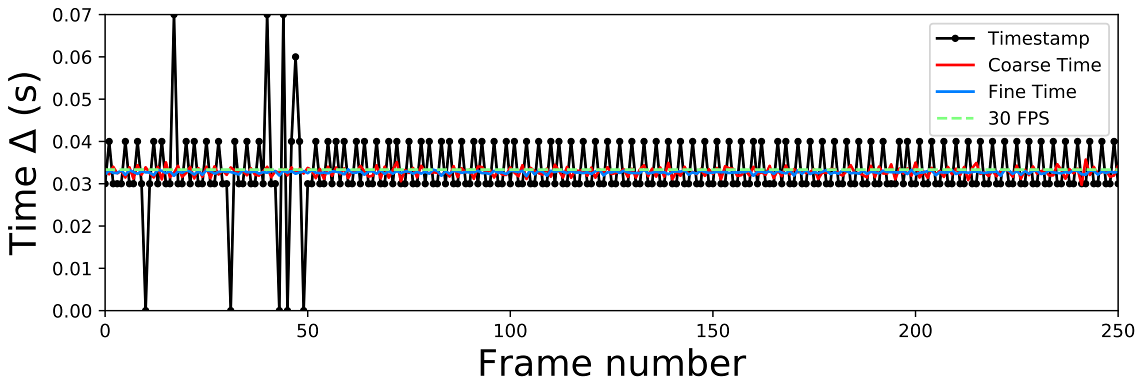


Figure J.9: Time deltas between frames according to camera frame timestamps (black), coarse (red) and fine (blue) rotor clocks, compared to nominal framerate (green). Reported clock timestamps fluctuate between 0.03 and 0.04 s (with occasional skipped or doubled frames resulting in anomalous deltas). Rotor clock times show that the actual recording times for each frame adhere much more closely to nominal framerate.

Skipped/Doubled Frames: Lastly, Figure J.10 shows an example of a doubled frame from camera p1c2. The same frame and timestamp are reported from the camera twice (frames 453 and 454). In frame 455, objects have moved 7-8 feet from their positions in frames 453 and 454, consistent with a time difference of roughly 2/30s (0.066s) for the average speeds at which the pictured vehicles are traveling, but the reported time delta is only 0.04s. This suggests that an inaccurate timestamp has been reported for frame 455.

J.2 Timestamp Error Correction

To correct these timestamp errors, we perform 2 main operations and consider a third.

Time Offset Correction

First we attempt to correct the camera clock offset or bias as we expect this value to be mostly fixed within a relatively short (1-2 min) time-frame. Let the estimate of the true time of frame j for camera k be denoted as $t'_{j,k}$, the reported camera timestamp be $t_{j,k}$the clock offset for camera k be denoted as o_k, and the *residual error* due to quantization and skipped/doubled frames for camera k and frame j denoted as ε_{jk}. We estimate the true time as:

$$t'_{j,k} = t_{j,k} + o_k + \varepsilon_{j,k} \tag{22}$$

Inspired by work on camera synchronization in sports [367], we use vehicles viewed simultaneously by two cameras with overlapping fields of view to estimate the relative clock offset between the two (o_k - $o_k - 1$). Let i index the set of m objects visible in a portion of the roadway coordinate system $[x_{i,min}, x_{i,max}]$ in both cameras k and $k-1$. We sample s x-coordinates (indexed by r) uniformly spaced across the range $[x_{i,min}, x_{i,max}]$. For each point in this range x_r, we linearly interpolate between the two closest annotations for camera k to estimate the time at which camera k would have reported object i in position x_r. Let $\tau_{r,k}$ denote this time. We set $o_k - o_{k-1}$ to be the mean difference between estimated times of sample x-points from the two cameras.

$$o_k - o_{k-1} = \frac{\sum_i \sum_r (\tau_{r,k} - \tau_{r,k-1})}{m * s} \tag{23}$$

and set o_0 = 0, allowing us to sequentially solve for the rest of camera clock errors. We ignore the effect of quantization for purposes of estimating the camera clock offsets. We assume that the per-camera clock offset does not drift over the relatively short duration of a single scene.

J.2.1 Residual Time Error Correction

Next we address the remaining timestamp errors caused by quantization and skipped/double frames. To reason about out-of-phase camera frames, we utilize a continuous functional representation of each vehicle's trajectory. Inspired by the work of [116] and [50], we fit a cubic spline to each of the x position and the y position of vehicle annotations as a function of reported timestamp, which ensures that the resulting trajectory follows a constant jerk (3rd-derivative) model between spline *knots* (points where polynomial coefficients of the spline function change). The number of knots k_i for is constrained based on reasonable driving assumptions according to equation 24:

$$k_i \leq 2d_i \tag{24}$$

where d_i is the difference between the maximum and minimum timestamps for which an annotation for vehicle i exists.

The best-fit spline $f_{x,i}(t)$ for the x coordinates of object i is found by minimizing the mean squared error between the spline-estimated object location and the annotated object location over all j annotated boxes for object i in all camera views:

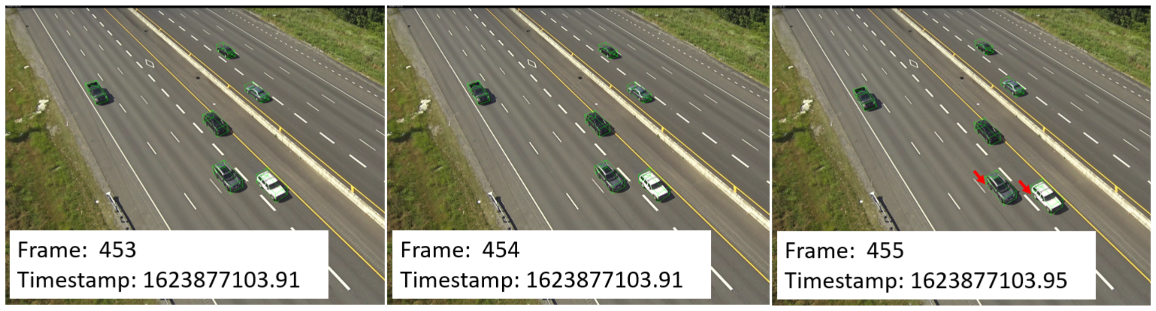


Figure J.10: Example of a doubled frame. Frame indices 453 and 454 for camera p1c2 in sequence 0 display identical frames and timestamps. The following frames shows object positional changes (red arrows) consistent with a larger time delta than is indicated by frame timestamps.

$$\min_{f_{x,i}} \sum_j (w_{i,j}(f_{x,i}(t_j) - x_i, j))^2 \tag{25}$$

where $w_{i,j}$ is a per-annotation weighting factor equivalent to the shift in pixels resulting from a one foot change in the initial x-position of the annotation. This weighting enforces that the resulting spline distorts visual object positions within the frame as little as possible. A similar equation is used to determine the best fit spline for y coordinates $f_{y,i}(t)$ as a function of time.

We then use the best-fit spline $f_i^{(x)}(t)$ for x-position for each object i visible within frame j of camera k to estimate the residual error for each camera frame $\varepsilon_{j,k}$ according to Equation 26:

$$\min_{\varepsilon_{j,k}} \sum_i (w_{i,j,k}(f_i^{(x)}(t_{j,k}) - x_{i,j,k}))^2 \tag{26}$$

where $w_{i,j,k}$ is the change (in pixels) resulting from a 1-foot change in x-position for object i in frame j of camera k. We constrain $\varepsilon_{j,k}$ to be at most the change associated with a skipped frame (1/30s) plus quantization error (0.01s).

J.2.2 Annotation shifting

Thus far, adjustments have only altered timestamps. Further enhancements to smooth vehicle trajectories as they travel through multiple camera fields of view can be made by slightly altering the raw annotations at the expense of annotation accuracy within a frame. However, noting that the RMSE for human annotations is 6.24 pixels (see next section), we consider small adjustments to raw annotations within this range. We test allowable shifts of 1,2, and 3 pixels in the x and y direction separately such that each annotation is moved slightly towards the best-fit spline position at the corresponding time. For each allowable maximum shift, the average shift performed is much smaller (0.25px, 0.36px, 0.42px and 0.47px, respectively). Note that we ultimately chose not to use any annotation-shifting for the released dataset, prioritizing accuracy within each frame over cross-camera trajectory smoothness.

J.3 Post-Correction Error Characterization

We characterize each known source of error in the dataset. When applicable, we characterize the error before and after timestamp corrections.

Human annotator variance is estimated by labeling the same object multiple times for a selection of vehicles. The root mean-squared error (RMSE) is computed for all single-vehicle annotations, and this metric is averaged across all sampled vehicles to estimate annotator root mean-squared error $RMSE_{ann}$ = 6.24 pixels for 4K resolution frames (3840×2160 pixels).

Vehicle size accuracy is verified based on known vehicle (make and model) sizes. A subset (10%) of labeled vehicles within the dataset are selected with discernible make and model. The annotated dimensions for each are compared to the actual size of that vehicle type when available, and otherwise to average vehicle size metrics for that class (semis and trucks). Over all sampled vehicles, a mean dimension error of -0.5ft, -0.1ft, and -0.2ft in length, width and height, respectively (standard deviation of 1.1ft, 0.3ft, and 0.6ft respectively). Mean errors for each dimension indicate only slight annotator bias towards undersized annotations, likely due to the ambiguous size of a curved 3D vehicle (vehicle classes with hard corners such as semis and trucks were not under-estimated). Full size comparison data is included in Appendix K.

Homography misalignment is estimated by comparing the labeled positions of the same roadway point in multiple camera fields of view. Average across all camera fields of view, the average cross-camera projection error for homography matching points is 0.54ft/0.18ft (x/y directions) and 15.6 pixels for 4K video.

Cross-camera vehicle annotation misalignment is computed by comparing vehicle annotations simultaneously visible in two or more cameras. We average the misalignment across all such annotations. Table 7 reports the cross-camera vehicle annotation misalignment in the x (CC_x) and y (CC_y) directions (in feet) and the cross-camera pixel error (CC_p) after each correction. The corresponding errors for the points used to define the homography themselves are also displayed; it is unlikely that any correction could reduce error below that threshold without adjusting raw annotations since this error is added to the annotations during homography transformation.

Correction	$CC_x \downarrow$	$CC_y \downarrow$	$CC_p \downarrow$
Homography	0.54 ft	0.18 ft	7.8 px
No Correction	1.52 ft	0.69 ft	35.2 px
Curve	1.52 ft	0.44 ft	29.6 px
Offset	1.39 ft	0.44 ft	22.2 px
Residual	1.24 ft	0.44 ft	15.6 px
1px Shift	1.08 ft	0.26 ft	11.8 px
2px Shift	0.98 ft	0.14 ft	9.2 px
3px Shift	0.91 ft	0.08 ft	7.0 px

Table 7: All measured metrics for raw annotations and each sequential correction. Pixel errors are for boxes drawn on 4K-resolution frames (3840×2160 pixels).

K Vehicle Size Estimation Error Data for I-24 3D Dataset (Ch. 5)

Table 8 provides all data used to compute vehicle dimension accuracy statistics. Vehicle sizes were obtained from manufacturer websites when possible, or else estimated from class size averages (trucks and semis). Vehicle make, model and year estimated, with some mistakes in model and year possible due to difficulty in estimating this information from imagery. Vehicle size estimates were used from a 10% subset of data selected across all dataset scenes.

			Annotation (ft)			True (ft)		
ID	Class	Vehicle Guess	L	W	H	L	W	H
98	midsize	2018 Kia Soul	13.3	5.6	5.4	13.6	5.9	5.3
94	sedan	2014 Toyota Prius	13.9	5.3	4.5	14.7	5.8	4.9
0	sedan	2003 Chevrolet Impala	15.9	5.7	4.5	16.7	6.1	4.8
1	pickup	2017 Toyota Tacoma	17.2	5.8	6.0	17.7	6.2	6.0
64	midsize	2012 Toyota 4Runner	16.1	6.0	5.8	15.8	6.3	6.0
68	midsize	2018 Honda Odyssey	16.4	6.5	5.9	16.9	6.6	5.8
34	semi	-	73.8	9.0	13.3	72.0	8.5	13.5
70	semi	-	76.1	8.6	14.8	72.0	8.5	13.5
80	midsize	2018 Hyundai Tuscon	14.6	6.2	6.1	14.7	6.1	5.4
81	pickup	2015 Nissan Frontier	17.5	6.3	5.7	17.2	6.1	5.8
88	midsize	2004 GMC Yukon	15.8	6.0	6.0	16.6	6.6	6.4
96	midsize	2020 Toyota RAV4	15.3	5.9	6.0	15.2	6.1	5.8
97	midsize	2020 Toyota RAV4	14.6	5.6	5.7	15.2	6.1	5.8
83	midsize	2012 Honda CR-V	14.4	5.9	4.9	14.8	6.0	5.4
85	pickup	2016 Nissan Titan	18.0	6.5	5.8	20.3	6.7	6.4
40	midsize	2014 Chevrolet Equinox	14.0	6.0	5.0	15.7	6.1	5.5
41	midsize	2014 Toyota Highlander	15.2	6.4	5.0	15.9	6.3	5.7
39	sedan	2013 Honda Accord	14.9	5.8	4.0	15.8	6.1	4.8
34	semi	-	73.8	9.0	13.3	72.0	8.5	13.5
32	midsize	2014 Dodge Caliber	13.6	6.0	4.9	14.5	5.8	5.0
28	sedan	2017 Honda Accord	15.1	6.0	4.4	15.8	6.1	4.8
17	van	2010 Chevrolet Express	18.0	6.5	6.9	18.7	6.6	6.9
15	pickup	2016 Ford F150	18.0	6.5	6.1	19.3	6.7	6.3
57	sedan	2015 fiat 500	11.0	5.4	4.6	11.7	5.3	4.9
58	truck	17 foot U-Haul style Box Truck	23.3	8.4	9.6	23.9	7.7	10.0

Appendix K is adapted from [95].

74	midsize	2015 Chrysler Town and Country	15.5	6.3	5.0	16.9	6.6	5.7
9	truck	20 foot U-Haul style Box Truck	26.8	8.0	13.3	26.6	7.7	10.1
87	midsize	2012 Honda CR-V	14.4	5.7	4.9	14.8	6.0	5.4
90	sedan	2014 Nissan Altima	15.4	6.1	4.1	16.0	6.0	4.8
63	truck	17 foot U-Haul style Box Truck	25.8	8.3	11.9	23.9	7.7	10.0
106	sedan	2004 Cadillac Deville	16.6	5.7	4.3	17.3	6.3	4.8
108	van	2012 Chevrolet Express	19.2	6.5	6.3	18.7	6.6	6.8
109	midsize	2012 Honda CR-V	14.2	5.5	5.0	14.8	6.0	5.4
113	pickup	2014 Dodge 1500	18.7	6.5	5.8	19.1	6.6	6.3
95	semi	-	72.1	8.9	12.8	72.0	8.5	13.5
26	van	2012 Chevrolet Express	17.8	6.5	7.0	18.7	6.6	6.8
27	midsize	Jeep Grand Cherokee	13.8	5.8	5.6	15.8	6.3	5.8
1	van	2018 Ford Transit 250	20.6	6.3	9.9	22.2	6.8	9.1
0	pickup	2018 Toyota Tacoma	18.0	6.5	6.5	18.8	6.3	6.0
7	semi	-	72.6	8.7	13.1	72.0	8.5	13.5
73	midsize	2018 Nissan Rogue	14.8	6.3	5.4	15.4	6.0	5.7
13	midsize	2004 Honda CR-V	13.0	5.4	6.3	14.9	5.8	5.5
93	midsize	2018 Kia Soul	12.4	6.0	5.0	14.3	5.8	5.3
257	van	2007 GMC Savana	19.2	6.3	6.8	18.7	6.6	6.8
34	midsize	2015 Chevrolet Suburban	17.2	6.7	5.5	18.5	6.6	6.4
248	pickup	2018 Chevrolet Silverado 1500	19.2	6.4	6.0	20.0	6.7	6.2
67	sedan	2016 Kia Forte	14.0	5.8	4.5	15.0	5.8	4.7
117	midsize	2014 Chevrolet Equinox	14.4	5.9	5.2	15.7	6.1	5.5
116	semi	-	73.5	8.2	13.7	72.0	8.5	13.5
64	sedan	2012 Nissan Altima	14.6	5.9	4.3	15.9	5.9	4.8
32	midsize	2016 Jeep Wrangler (4 door)	14.2	6.3	5.5	15.3	6.2	6.1
114	midsize	2015 Dodge Grand Caravan	16.0	6.0	5.5	16.9	6.6	5.8
4	sedan	2014 Toyota Prius	13.8	5.4	4.5	14.7	5.8	4.9
90	midsize	2018 Honda Fit	12.8	5.4	4.5	13.4	5.6	5.0
0	van	2016 Dodge Sprinter	18.2	6.4	7.5	19.4	6.7	7.8
10	truck	17 foot U-Haul style Box Truck	24.8	6.8	10.4	23.9	7.7	10.0
245	midsize	2012 Chevrolet HHR	13.6	5.4	5.0	14.7	5.8	5.3
238	van	2015 Ford Transit	17.2	6.3	7.1	18.3	6.8	7.0
98	sedan	2017 Chrysler 300	15.3	6.2	4.2	16.6	6.3	4.9
140	truck	17 foot U-Haul style Box Truck	24.0	7.9	9.5	23.9	7.7	10.0
183	semi	-	72.8	8.8	12.7	72.0	8.5	13.5
101	midsize	2018 Honda Fit	12.2	5.2	4.7	13.4	5.6	5.0
182	midsize	2014 Chevrolet Equinox	14.2	6.2	5.1	15.7	6.1	5.5
95	sedan	2015 Chevrolet Malibu	15.0	6.1	4.2	16.0	6.0	4.8
137	sedan	2017 Ford Fiesta	12.2	5.3	4.7	13.3	5.7	4.8
94	sedan	2014 Toyota Corolla	14.8	6.0	4.2	15.3	5.8	4.8
92	sedan	2015 Chevrolet Cruze	14.2	5.8	4.2	15.1	5.9	4.8
176	sedan	2016 Nissan Leaf	11.2	5.4	4.6	14.6	5.8	5.1
134	pickup	2016 Chrevolet Silverado 1500	18.0	6.6	5.3	17.1	6.7	6.2
214	midsize	2018 Jeep Compass	13.4	5.7	5.0	14.4	6.2	5.4
224	midsize	2020 Toyota RAV4	14.0	6.2	5.3	15.2	6.1	5.8
229	pickup	2016 Ford F150 Crew Cab	18.6	6.8	5.7	19.3	6.7	6.3

Table 8: Annotated and true dimensions for assessed data subset (10% of vehicles).

Table 9 reports aggregate size estimate error metrics for each class and averaged over all samples. For each dimension (length, width and height) the percentage of vehicles that are reported with less than 1 foot of error in this dimension is summarized, as is the mean error and standard deviation in annotations errors). Over all sampled vehicles, a mean dimension error of -0.5ft, -0.1ft, and -0.2ft in length, width and height, respectively, was obtained (annotations were too small on average). The standard deviation for each error was 1.1ft, 0.3ft, and 0.6ft respectively. Mean errors for each dimension indicate only slight annotator bias towards undersized annotations, likely due to the difficult of exactly sizing curved 3D vehicles from an oblique angle (notably, vehicles with hard corners such as semis and trucks were not under-estimated). The dimension with the largest distribution of error was length (1.1ft standard deviation). Notably, 96% of vehicle annotations have height accurate within 1 foot and 100% of vehicle annotations have width accurate within 1 foot.

Class	Samples	Length (ft)	Width (ft)	Height (ft)
sedan	16	-1.0 (0.7)	-0.2 (0.2)	-0.5 (0.2)
midsize	28	-0.9 (0.6)	-0.2 (0.3)	-0.3 (0.4)
van	7	-0.6 (0.7)	-0.2 (0.2)	+0.0 (0.5)
pickup	9	-0.6 (0.9)	-0.1 (0.2)	-0.3 (0.4)
truck	5	+0.5 (0.9)	+0.2 (0.6)	0.9 (1.6)
semi	7	+1.5 (1.3)	+0.2 (0.3)	-0.1 (0.7)
Under 1 ft Error	-	60%	100%	96%
Total	**72**	**-0.5(1.1)**	**-0.1 (0.3)**	**-0.2 (0.6)**

Table 9: Annotation dimension error mean (standard deviation) compared to known vehicle sizes. + indicates mean estimated dimension is too large.

L Additional Experimental Settings and Implementation Details for I-24 3D Dataset (Ch. 5)

L.1 Evaluation Protocol

To assess the difficulty of the tracking dataset and to provide initial evidence on the suitability of existing tracking algorithms, we benchmark a set of tracking methods on this dataset. Experimental protocol, metrics for evaluation, and implemented algorithms are described in this Appendix.

L.1.1 Model Training

Each scene is split into temporally contiguous training and validation partitions (the first 80% and the last 20% of each scene, respectively). Thus, the validation partition of Scene 1 consists of all frames greater than or equal to 2160 from each sequence, and the validation partition consists of all frames greater than or equal to 1440 for Scenes 2 and 3. Detection model training is performed exclusively using the training partition. All training is performed locally on RTX6000 GPUs, and detection models are trained until convergence (Generally 10-15 epochs).

L.1.2 Tracking

Camera frames are out of phase (see Appendix J). To account for this, during tracking we maintain tight 1/60th second synchronization between videos during tracking using corrected frame timestamps, skipping frames as necessary to nominally maintain a 15 Hz frame rate (empirically, performance degrades above this frame rate due to increased false positives). We perform tracking across all video sequences for a scene in parallel (that is, all detections and tracklet updates are performed across all cameras for the same approximate time). This is required for crop-based tracking pipelines [91] but not for tracking-by-detection pipelines. Tracking is performed through the entire scene duration (i.e. across the training and validation partitions of the dataset).

Appendix L is adapted from [95].

L.1.3 Detector AP Testing

We evaluate each detector on the validation partition of the I24-3D dataset. We follow the procedure detailed in The Pascal VOC Dataset Challenge guidelines [6], including penalizing for multiple predicted objects corresponding to the same ground truth object. To filter objects, we remove all objects with output confidence less than 0.01, and perform non-maximal suppression on the set of detection outputs, first in the image coordinate system with a requisite IOU of 0.4 for removal, then in roadway coordinates with a requisite IOU of 0.1 for removal. We compare the remaining set of detections against the ground truth detections in roadway coordinates. We use *birds-eye view precision* (AP_{bev}) rather than *3D precision* (AP_{3D}) (i.e. height is not included in the evaluation).

L.1.4 Tracking Evaluation

We evaluate each tracking pipeline on each entire scene, including both the training and validation partitions. We fit a best-fit 3rd order polynomial spline to each ground truth object to obtain a continuous object representation in roadway coordinates as described in Appendix I. Predicted vehicle trajectories are compared against boxes sampled from the best-fit spline for each object. Since ground truth objects are labeled in cameras with varying start times and tracked objects are produced with synchronized timestamps, we compute and compare only the temporally-overlapping sections of each ground truth trajectory dataset with the predictions. We linearly interpolate between the spline-sampled boxes and the tracker-output predictions at 30Hz to produce object sets at the same discrete times.

IOU Threshold: For all metrics except HOTA (which uses a variable IOU threshold for considered matches), we use a required IOU of 0.3 throughout evaluation. This is because, despite our best efforts to fully rectify annotations corresponding to the same vehicle viewed from different cameras at the same time, we are not able to fully remove the projection errors between these annotations (see Appendix J). Thus, we seek to avoid penalizing tracking algorithms for output errors that could reasonably be an artifact of these inconsistencies, so we select a somewhat lax IOU threshold to account for 1.24ft/0.44ft X/Y cross-camera annotation misalignment.

L.2 Algorithm Implementation Details

L.2.1 3D Detectors

- **Monocular 3D Detector (Single3D)** - a Retinanet model with Resnet34-FPN backbone [132]. The outputs from this network are parameterized as a rectangular prism rather than as corner coordinates, which empirically leads to better model convergence. The formulation is camera-agnostic (as training a separate model for each camera FOV is infeasible both from data scarcity and scalability standpoints.) We remove all detections with a confidence lower than 0.3, and perform non-maximal suppression on detection outputs per camera, in pixel coordinates, with an IOU threshold of 0.4, then in shared roadway coordinates with an IOU threshold of 0.01. We quantize the model to half precision (16-bit float) for speed at inference. Code is originally from https://github.com/yhenon/pytorch-retinanet.
- **Monocular 3D Multi-frame Detector (Dual3D)** - Inspired by recent works utilizing multiple frames for detection and tracking [87], we add the previous frame as detection input. We double the input channels of the model's first convolutional layer to accomodate the additional input. As above, we remove all detections with a confidence lower than 0.3, and perform non-maximal suppression on detection outputs per camera, in pixel coordinates, with an IOU threshold of 0.4. We quantize the model to half precision (16-bit float) for speed at inference.
- **Monocular 3D Crop Detector (CBT)** - as described in [91], we train a Retinanet Model with Resnet34-FPN backbone for detecting objects in cropped portions of full frames. We expand each object prior by 1.3x to select the relevant pixels of a frame for each object, and resize these pixels to a crop size of 112x 112 pixels. After detection, we keep only the 50 highest confidence outputs from the detector and then using a weighting factor W of 0.4 to weight confidence and IOU with object prior to select the best detection for each existing object. On full-frame detections frames (every 4 frames), we use the Dual3D detector.
- **Ground Truth Detections (GT)** - perfect ground-truth detections, stored natively in roadway coordinates.

L.2.2 Object Trackers

- **Kalman-Filter IOU Tracker (KIOU)** - as described in [166]. We utilize a contant velocity roadway-coordinate Kalman filter for object position prediction. We use the object-to-detection intersection over union metric in roadway coordinates to select the best-matching detection for each existing object.
- **ByteTracker (Byte)** - we utilize the two-stage association method described in [226], using IOU as both primary and secondary matching criterion and utilizing a Kalman filter as suggested by authors. As suggested by the authors, we relax the criteria of each detector such that all objects with confidence higher than 0.3 are kept for the primary matching step and all object with confidence between 0.01 and 0.3 are kept for the secondary matching phase.
- **Crop-based Tracking (CBT)** - as proposed in [91], detection on some frames is performed by re-detecting priors in cropped subsets of the overall frame, and object associations are implicit for these frames.
- **Ground Truth Single Camera Tracklets** - perfect single-camera tracklets.

L.2.3 Cross-Camera Rectification Method

- **Detection Fusion (DF)** - as preferred in the AV context [285], detections from all cameras are combined online in roadway coordinates and non-maximal suppression with a stringent 0.01 IOU threshold utilized to eliminate overlapping detections.
- **Trajectory Stitching (TF)** - as proposed in [311], single camera tracklets are compared for spatio-temporal overlap offline, stitched together when a matching criteria is met, and refined to optimally describe the observed set of tracked object positions. We refer the interested reader to the cited work for an explanation of parameters and their meanings. We use this algorithm as implemented at https://github.com/yanb514/I24-postprocessing.
- **None** - as a baseline, object tracklets from each camera are output with no fusion.
- **Both (DF+TF)** - Tracking uses detection fusion, and a subsequent trajectory stitching step is performed to deal with remaining object fragmentations.

M Full Results for I-24 3D Dataset (Ch. 6)

This Appendix details AP testing results, tracking results for pipelines utilizing ground truth detections or tracklets as input, and finally lists per-scene results for all pipelines.

M.1 Detector AP Testing Results

Figure M.11 show the results of detector average precision testing. At all tested IOU thresholds, the Dual3D network has the highest AP score (0.572 AP_{70}), and the Single3D model has the lowest AP score at all 3 thresholds (0.254 AP_{70}). Interestingly, despite the large difference in detection precision, comparable tracking pipelines using these two detectors show only slight or negligible performance difference, perhaps explainable by insufficient Kalman filter parameter tuning to account for the more accurate measurements provided by the Dual3D detector.

Appendix M is adapted from [95].

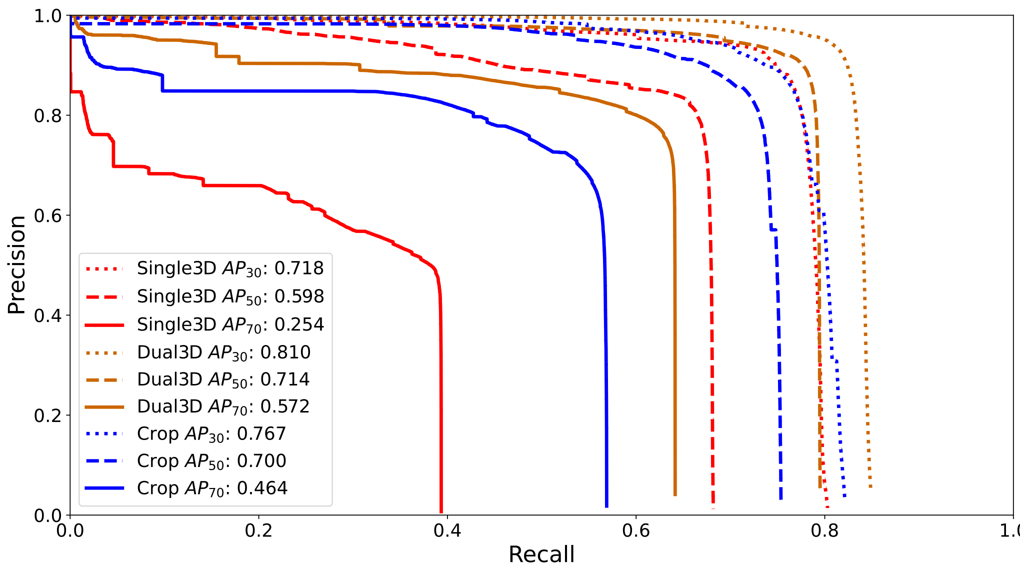


Figure M.11: Precision versus recall curves for each detector at IOU thresholds of 0.7 (solid), 0.5 (dash) and 0.3 (dot), generated as in [6]. Overall AP score for each model at each threshold is listed in the legend.

M.2 Results for Ground Truth Pipelines

Table 10 shows per-scene and average results for each implemented pipeline using ground truth detections as input. A few results are of note. Trajectory Fusion (TF) is the most accurate multi-camera rectification method, with Detection Fusion (DF) and DF + TF slightly less accurate and roughly equal across both trackers (KIOU and ByteTrack). Though recall is high (89%) and precision is very high (as high as 99.8 %), still only 73.6% of objects are fully tracked at best. These results show that high-quality detections alone are not enough to solve the multi-camera tracking problem.

Table 11 shows results for all pipelines utilizing ground truth single-camera tracklets, either as-is or with a subsequent trajectory fusion (TF) step. With trajectory fusion, an HOTA of 61.7% is achieved. This score is mostly driven down by remaining ID switches (0.51 per ground truth object on average), as the localization accuracy (MOTP) is fairly high (83.6% average). Intuitively, it makes sense that given great single-camera tracklets, the remaining difficulties are caused by ID switches across cameras which constitute failures in trajectory fusion. The results of the AP testing, ground truth detection and tracklet pipelines indicate that there is room for improvement in all 3 components of the implemented multi-camera pipelines (detector, tracker and multi-camera rectification).

Tra.	DF	TF	Scene	HOTA	MOTA	MOTP	Rec	Prec	GT%	Pred%	MT	ML	Sw/GT
Byte	✓	✓	1	66.5	94.0	78.5	94.3	99.7	96.6	100.0	89.8	0.9	0.13
Byte	✓	✓	2	47.3	85.6	62.2	88.0	97.4	86.0	100.0	69.3	3.5	0.23
Byte	✓	✓	3	40.9	76.3	82.6	79.1	96.8	92.5	99.8	57.7	5.3	1.30
Byte	✓	✓	*avg*	51.6	85.3	74.4	87.1	98.0	91.7	99.9	72.2	3.3	0.55
KIOU	✓	✓	1	66.3	92.2	78.5	93.2	99.0	96.0	100.0	89.8	2.2	0.11
KIOU	✓	✓	2	48.2	87.0	62.0	89.0	97.9	86.0	100.0	73.7	4.4	0.20
KIOU	✓	✓	3	38.3	65.7	69.9	74.8	89.2	82.6	93.0	44.5	19.9	0.59
KIOU	✓	✓	*avg*	50.9	81.7	70.1	85.7	95.4	88.2	97.7	69.3	8.8	0.30
Byte		✓	1	71.7	95.5	78.5	95.5	99.9	96.6	100.0	91.9	0.9	0.00
Byte		✓	2	69.2	89.3	86.3	90.4	98.8	86.0	100.0	68.4	3.5	0.22
Byte		✓	3	37.8	73.1	82.3	80.8	91.5	93.6	95.8	60.1	4.3	1.64
Byte		✓	*avg*	**59.6**	**86.0**	82.4	88.9	96.8	**92.0**	98.6	73.5	**2.9**	0.62
KIOU		✓	1	71.7	95.5	78.5	95.5	99.9	96.6	100.0	91.9	0.9	0.00
KIOU		✓	2	68.7	89.1	86.3	90.2	98.9	86.0	100.0	67.5	3.5	0.23
KIOU		✓	3	38.1	73.4	82.3	81.4	91.3	93.6	95.4	61.2	4.3	1.63
KIOU		✓	*avg*	59.5	**86.0**	82.4	**89.0**	96.7	**92.0**	98.5	**73.6**	**2.9**	0.62
Byte	✓		1	63.0	91.3	90.9	91.5	100.0	96.6	100.0	85.4	0.9	0.69
Byte	✓		2	52.4	82.8	92.0	83.4	99.5	86.0	100.0	58.8	3.5	1.02
Byte	✓		3	38.8	76.7	91.6	77.0	100.0	93.2	100.0	52.3	5.3	2.17
Byte	✓		*avg*	51.4	83.6	91.5	84.0	**99.8**	91.9	**100.0**	65.5	3.3	1.29
KIOU	✓		1	62.7	91.4	91.0	91.6	100.0	96.6	100.0	85.4	0.9	0.71
KIOU	✓		2	51.4	82.6	92.0	83.3	99.4	86.0	100.0	57.0	4.4	1.09
KIOU	✓		3	39.3	76.7	91.7	77.0	100.0	93.2	100.0	50.9	6.0	2.10
KIOU	✓		*avg*	51.1	83.6	**91.6**	83.9	**99.8**	91.9	**100.0**	64.4	3.8	1.30
Byte			1	23.6	73.5	90.1	91.1	85.4	96.6	97.6	85.1	0.9	9.40
Byte			2	28.2	68.5	91.6	83.7	85.3	86.0	86.8	60.5	3.5	4.93
Byte			3	24.9	67.9	91.3	79.3	88.1	93.6	89.9	54.1	4.3	6.33
Byte			*avg*	25.6	70.0	91.0	84.7	86.3	**92.0**	91.4	66.6	**2.9**	6.88
KIOU			1	23.6	73.5	90.1	91.1	85.4	96.6	97.6	85.1	0.9	9.40
KIOU			2	28.2	68.5	91.6	83.7	85.3	86.0	86.8	60.5	3.5	4.93
KIOU			3	24.9	67.9	91.3	79.3	88.1	93.6	89.9	54.1	4.3	6.33
KIOU			*avg*	25.6	70.0	91.0	84.7	86.3	**92.0**	91.4	66.6	**2.9**	6.88

Table 10: Results for all tracking pipelines using ground truth (GT) detections on each scene. Results include higher order tracking accuracy (HOTA), multiple object tracking accuracy / precision (MOTA/MOTP), recall (Rec), precision (Prec), ground truth and prediction match rates (GT% / Pred %), mostly tracked and mostly lost objects (MT/ML) and number of ID switches per ground-truth object (Sw/GT). Best average result for each metric across all pipelines shown in bold.

TF	Scene	HOTA	MOTA	MOTP	Rec	Prec	GT%	Pred%	MT	ML	Sw/GT
✓	1	59.5	76.2	78.5	88.2	88.1	100.0	99.1	82.6	0.9	0.16
✓	2	77.6	95.6	88.3	97.0	98.6	100.0	98.5	95.6	2.6	0.22
✓	3	47.9	70.6	84.1	95.7	79.3	99.3	92.1	96.4	1.1	1.16
✓	*avg*	**61.7**	**80.8**	83.6	93.6	**88.7**	99.8	**96.6**	91.6	1.5	**0.51**
	1	19.9	14.6	90.3	89.2	55.0	100.0	76.9	84.2	0.3	9.15
	2	29.3	38.7	92.4	95.1	63.0	100.0	83.0	98.2	0.0	5.25
	3	28.7	36.9	92.8	98.2	61.8	100.0	83.5	98.6	0.0	5.25
	avg	26.0	30.1	**91.8**	**94.2**	59.9	**100.0**	81.1	**93.7**	**0.1**	6.55

Table 11: Results for all tracking pipelines using ground truth single camera tracklets on each scene. Best average result for each metric across all pipelines shown in bold.

M.3 Per-scene Results for Tracking Pipelines

Tables 12, 13, and 14 report the results for each pipeline using the Dual3D, Single3D, and Crop3D detectors, respectively, on each scene. Across most pipelines, Scene 3 is the most difficult and Scene 1 is the easiest.

On Scene 1, the best performing pipeline (Dual3D + KIOU + TF) achievs 58.5% HOTA and 86.3% mostly tracked objects, still not accurate enough for fine-grained traffic analyses (HOTA 0.75 and 95% mostly tracked objects). The best-performing pipelines for Scene 3 (Single3D or Dual3D + KIOU + TF) acheive just 29.1% HOTA). All pipelines utilizing Crop3D perform poorly on Scene 2 (best HOTA 11.5%). This is because Crop3D searches within a local region around each object prior, and always utilizes the best detection from this local crop to update the object's position. This strategy fails when the region is occluded (e.g. by snow) which is often the case in Scene 2.

Finally, Figure M.12 shows the HOTA curves for the best performing pipeline (Dual3D + KIOU + TF) relative to the baseline with no cross-camera rectification, for Scene 1. *Detection* and *Association* scores refers to the DetA formula and AssA defined in [218], which are roughly meant to appraise the accuracy of the detection and object matching performance of the tracker independently. *Higher Order Tracking Accuracy* (HOTA) is an aggregate metric composed of these two components, evaluated at 19 evenly spaced IOU thresholds required for a prediction to be considered a true positive. Lower thresholds result in higher scores because more predictions are deemed valid matches according to the threshold. Note that for both pipelines, the association score is lower than the detection score, indicating that more the cross-camera association problem is more problematic than detection accuracy for achieving high HOTA. Additionally, it can be seen that performance of all metrics declines steeply at a required IOU of 0.5 and higher, meaning that more precise object localization could likely also improve HOTA scores considerably.

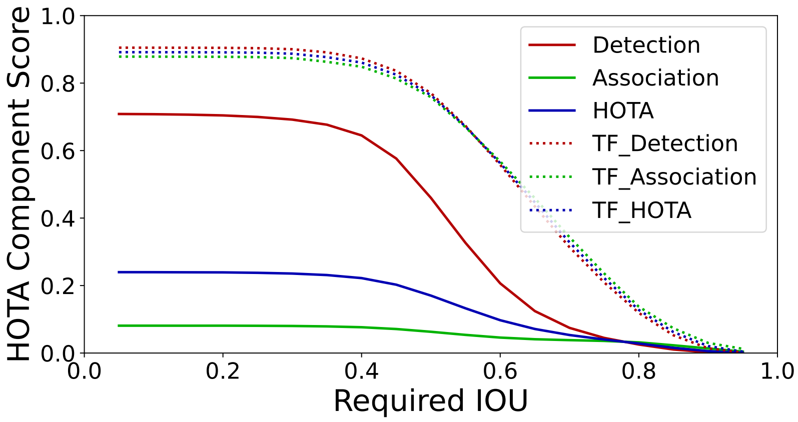


Figure M.12: *Detection* accuracy, *Association* accuracy, and HOTA as defined for Dual3D + KIOU pipeline (Solid) and Dual3D + KIOU + TF (dotted).

Tra.	DF	TF	Scene	HOTA	MOTA	MOTP	Rec	Prec	GT%	Pred%	MT	ML	Sw/GT
Byte	✓	✓	1	25.9	19.2	49.0	54.1	60.8	88.5	91.2	26.4	25.2	0.17
Byte	✓	✓	2	43.2	74.8	73.9	80.8	93.2	91.2	89.7	55.3	8.8	0.57
Byte	✓	✓	3	23.6	56.0	68.3	61.9	91.5	92.2	99.3	26.0	11.0	1.98
Byte	✓	✓	*avg*	30.9	50.0	63.7	65.6	81.9	90.6	93.4	35.9	15.0	0.91
KIOU	✓	✓	1	52.1	86.2	68.9	89.3	96.6	95.0	99.5	80.1	3.1	0.19
KIOU	✓	✓	2	43.1	74.5	74.0	80.1	93.6	91.2	90.3	53.5	8.8	0.57
KIOU	✓	✓	3	24.0	54.0	64.5	60.2	90.9	88.3	96.1	23.8	19.2	1.29
KIOU	✓	✓	*avg*	39.7	71.6	69.2	76.5	93.7	91.5	**95.3**	52.5	10.4	0.68
Byte		✓	1	57.6	89.0	69.1	92.2	96.6	95.3	98.8	84.2	2.5	0.04
Byte		✓	2	29.8	73.1	62.8	78.8	93.4	93.9	86.1	52.6	7.0	1.30
Byte		✓	3	28.7	62.9	64.9	69.5	91.4	89.7	96.0	40.2	14.6	1.07
Byte		✓	*avg*	38.7	75.0	65.6	80.2	93.8	93.0	93.6	59.0	8.0	0.80
KIOU		✓	1	58.5	89.7	69.2	92.9	96.7	95.3	98.4	86.3	2.2	0.02
KIOU		✓	2	46.9	77.7	74.5	86.2	91.1	90.4	82.4	64.0	9.6	0.49
KIOU		✓	3	29.1	63.5	64.8	69.9	91.7	89.3	96.1	40.9	14.6	1.05
KIOU		✓	*avg*	**44.8**	**77.0**	**69.5**	**83.0**	93.2	91.7	92.3	**63.8**	8.8	**0.52**
Byte	✓		1	26.5	72.8	57.8	77.2	95.0	94.7	97.9	57.8	4.0	1.51
Byte	✓		2	18.7	53.2	70.3	59.0	91.7	91.2	87.3	14.0	9.6	3.50
Byte	✓		3	17.5	54.6	66.5	56.2	97.7	92.2	98.7	16.7	12.5	3.27
Byte	✓		*avg*	20.9	60.2	64.9	64.2	94.8	92.7	94.7	29.5	8.7	2.76
KIOU	✓		1	26.8	73.4	57.9	77.3	95.6	94.4	98.5	57.1	4.0	1.50
KIOU	✓		2	19.1	53.3	70.5	58.6	92.3	91.2	86.9	14.9	10.5	3.35
KIOU	✓		3	17.5	54.5	66.6	56.2	97.7	92.2	98.8	17.1	12.1	3.30
KIOU	✓		*avg*	21.1	60.4	65.0	64.0	**95.2**	92.6	94.7	29.7	8.9	2.72
Byte			1	14.1	60.5	57.9	82.1	80.7	94.7	94.7	66.5	3.1	8.57
Byte			2	15.6	50.1	69.4	68.6	79.5	91.2	79.1	25.4	8.8	6.66
Byte			3	15.2	54.8	66.8	67.6	84.8	**93.6**	88.7	35.6	8.5	6.57
Byte			*avg*	15.0	55.1	64.7	72.8	81.7	93.2	87.5	42.5	**6.8**	7.27
KIOU			1	14.2	61.3	58.1	82.5	81.0	94.7	94.9	67.4	3.4	8.63
KIOU			2	15.7	50.8	69.7	67.9	80.6	91.2	79.3	22.8	8.8	6.55
KIOU			3	15.2	54.8	66.9	67.6	84.9	93.2	89.0	36.7	8.9	6.57
KIOU			*avg*	15.1	55.6	64.9	72.7	82.2	93.1	87.8	42.3	7.0	7.25

Table 12: Results for all tracking pipelines using Dual3D detections on each scene. Best average result for each metric across all pipelines shown in bold.

Tra.	DF	TF	Scene	HOTA	MOTA	MOTP	Rec	Prec	GT%	Pred%	MT	ML	Sw/GT
Byte	✓	✓	1	28.1	26.6	51.5	58.0	64.9	90.7	91.7	33.9	23.6	0.19
Byte	✓	✓	2	29.9	66.8	65.8	70.9	94.6	95.6	86.7	34.2	5.3	1.16
Byte	✓	✓	3	24.4	54.5	65.2	59.6	92.3	90.0	96.9	21.0	17.4	1.25
Byte	✓	✓	*avg*	27.5	49.3	60.8	62.8	83.9	92.1	91.8	29.7	15.4	0.86
KIOU	✓	✓	1	52.3	86.1	69.1	89.4	96.5	96.3	99.5	81.7	1.9	0.17
KIOU	✓	✓	2	42.9	74.1	75.2	79.7	93.5	93.9	91.2	51.8	7.0	0.61
KIOU	✓	✓	3	24.5	54.5	65.2	59.7	92.2	90.0	96.3	21.4	17.4	1.24
KIOU	✓	✓	*avg*	39.9	71.6	69.8	76.3	94.1	93.4	**95.6**	51.6	8.8	0.67
Byte		✓	1	57.8	89.2	69.2	92.7	96.4	96.6	98.5	86.6	1.6	0.02
Byte		✓	2	31.9	73.5	65.2	78.8	93.8	95.6	85.5	50.0	4.4	1.22
Byte		✓	3	25.0	55.1	70.0	70.3	82.5	92.5	93.5	39.5	8.2	2.19
Byte		✓	*avg*	38.2	72.6	68.2	80.6	90.9	**94.9**	92.5	58.7	**4.7**	1.15
KIOU		✓	1	57.5	89.3	69.2	92.9	96.3	96.6	97.9	87.0	1.6	0.03
KIOU		✓	2	47.7	78.2	75.4	86.4	91.5	93.0	79.8	64.0	7.0	0.47
KIOU		✓	3	29.1	63.8	65.1	69.7	92.3	90.4	96.1	35.6	14.9	1.05
KIOU		✓	*avg*	**44.8**	**77.1**	**69.9**	**83.0**	93.4	93.3	91.3	**62.2**	7.8	**0.52**
Byte	✓		1	26.1	72.5	58.2	76.6	95.2	96.0	97.8	55.3	2.5	1.55
Byte	✓		2	19.3	54.2	70.9	59.0	93.0	93.9	86.0	11.4	7.9	3.31
Byte	✓		3	18.4	54.3	69.2	56.0	97.6	91.5	98.7	14.6	12.1	3.09
Byte	✓		*avg*	21.3	60.3	66.1	63.9	95.3	93.8	94.2	27.1	7.5	2.65
KIOU	✓		1	26.4	72.6	58.4	76.6	95.5	96.3	97.7	53.7	2.2	1.52
KIOU	✓		2	19.4	53.9	71.2	58.5	93.2	94.7	85.5	11.4	7.9	3.25
KIOU	✓		3	18.4	54.3	69.2	55.9	97.7	91.5	98.4	14.2	12.5	3.09
KIOU	✓		*avg*	21.4	60.3	66.2	63.7	**95.5**	94.2	93.9	26.5	7.5	2.62
Byte			1	14.0	59.9	57.9	81.6	80.5	96.6	94.1	64.6	1.9	8.51
Byte			2	15.9	50.9	69.8	68.8	80.1	93.9	76.7	24.6	6.1	6.48
Byte			3	15.4	51.1	69.0	66.5	81.9	92.5	87.0	32.4	9.3	6.49
Byte			*avg*	15.1	54.0	65.5	72.3	80.8	94.3	85.9	40.5	5.8	7.16
KIOU			1	14.2	61.0	58.1	82.2	81.0	96.6	94.4	65.5	1.6	8.53
KIOU			2	16.0	50.8	70.1	67.8	80.6	94.7	77.0	21.1	6.1	6.32
KIOU			3	15.4	51.4	69.0	66.5	82.2	92.2	87.1	31.7	9.3	6.49
KIOU			*avg*	15.2	54.4	65.7	72.2	81.3	94.5	86.1	39.4	5.6	7.12

Table 13: Results for all tracking pipelines using Single3D detections on each scene. Best average result for each metric across all pipelines shown in bold.

Tra.	DF	TF	Scene	HOTA	MOTA	MOTP	Rec	Prec	GT%	Pred%	MT	ML	Sw/GT
Byte	✓	✓	1	35.8	43.5	68.4	68.7	73.3	92.5	82.3	49.7	13.4	0.34
Byte	✓	✓	2	11.4	-27.9	66.3	31.7	34.8	90.4	44.6	6.1	40.4	1.89
Byte	✓	✓	3	23.5	48.2	65.3	59.8	84.0	88.6	91.8	21.0	21.4	1.20
Byte	✓	✓	*avg*	23.6	21.3	66.7	53.4	64.0	90.5	72.9	25.6	25.0	**1.14**
KIOU	✓	✓	1	36.7	46.5	68.7	71.0	74.4	93.2	81.7	50.0	10.6	0.35
KIOU	✓	✓	2	11.5	-34.0	68.2	30.5	32.2	90.4	38.2	4.4	39.5	1.70
KIOU	✓	✓	3	25.6	51.8	69.3	61.9	86.2	87.9	93.8	28.5	16.7	1.36
KIOU	✓	✓	*avg*	**24.6**	21.4	**68.7**	54.4	64.2	90.5	71.2	27.6	22.3	**1.14**
Byte		✓	1	13.2	-31.7	47.7	38.8	35.6	89.4	59.5	14.0	35.4	0.86
Byte		✓	2	9.1	-65.8	66.2	29.7	23.8	91.2	34.3	6.1	39.5	2.30
Byte		✓	3	23.3	47.9	65.5	62.1	81.6	90.4	85.5	23.5	21.7	1.36
Byte		✓	*avg*	15.2	-16.5	59.8	43.5	47.0	90.4	59.8	14.5	32.2	1.51
KIOU		✓	1	32.4	26.7	67.6	67.8	62.3	93.2	62.8	48.8	14.6	0.43
KIOU		✓	2	7.1	-74.0	61.3	26.2	20.8	89.5	26.9	3.5	41.2	2.67
KIOU		✓	3	22.8	40.9	69.4	60.8	75.4	87.9	71.5	24.2	23.5	1.53
KIOU		✓	*avg*	20.7	-2.1	66.1	51.6	52.9	90.2	53.7	25.5	26.4	1.54
Byte	✓		1	24.7	58.4	59.0	74.7	82.4	94.4	93.5	51.9	5.3	1.43
Byte	✓		2	10.9	-11.2	66.7	37.8	43.8	92.1	53.1	3.5	28.9	3.55
Byte	✓		3	22.0	56.9	68.7	62.4	92.2	89.3	97.7	26.0	15.7	2.22
Byte	✓		*avg*	19.2	**34.7**	64.8	58.3	**72.8**	**91.9**	**81.4**	27.1	16.6	2.40
KIOU	✓		1	25.2	59.0	59.1	75.6	82.3	95.0	94.1	51.2	4.3	1.42
KIOU	✓		2	10.5	-20.5	67.0	35.4	38.9	92.1	47.6	1.8	32.5	3.49
KIOU	✓		3	22.2	57.7	68.8	62.8	92.9	88.6	97.3	24.6	14.9	2.21
KIOU	✓		*avg*	19.3	32.1	65.0	57.9	71.4	**91.9**	79.7	25.9	17.3	2.37
Byte			1	22.0	45.1	59.0	76.4	71.2	94.1	81.9	56.8	5.0	1.90
Byte			2	9.2	-46.2	66.1	37.9	31.2	91.2	42.8	4.4	24.6	4.62
Byte			3	21.4	57.4	68.7	65.1	89.7	90.4	94.2	29.9	15.7	2.61
Byte			*avg*	17.6	18.7	64.6	**59.8**	64.0	**91.9**	73.0	**30.4**	**15.1**	3.04
KIOU			1	22.5	47.8	59.2	77.1	72.8	95.0	83.0	58.4	4.3	1.89
KIOU			2	8.2	-62.8	66.0	33.4	25.9	90.4	39.7	1.8	28.9	5.09
KIOU			3	20.0	47.5	69.4	62.0	81.3	90.4	75.5	24.6	22.1	2.74
KIOU			*avg*	16.9	10.8	64.9	57.5	60.0	**91.9**	66.0	28.2	18.5	3.24

Table 14: Results for all tracking pipelines using Crop3D detections on each scene. Best average result for each metric across all pipelines shown in bold.

N Privacy Considerations for I-24 3D Dataset

As with any dataset containing video data of a public location, the I24-3D dataset potentially contains *personally identifiable information* (PII). We visually inspect video sequences to ensure that license plates are not visible at a visually distinctive level (license plate numbers cannot be determined from imagery except possibly with extensive de-noising techniques). We further attempt to process each video sequence with a license plate blurring software, but the software detects almost entirely false positives indicating that plate information is not visually discernable. Likewise, we confirm that driver faces in each vehicle are not visually discernible, no pedestrians are visible within the dataset, and no anomalous events (e.g. crashes) occur. Lastly, we have submitted this research to University *Institutional Review Board* (IRB) and secured research approval to ensure that the dataset management protocols appropriately protect individuals' privacy.

O Coordinate Systems for the I-24 Video Dataset (Ch. 7)

Appendix N is adapted from [95].
Appendix O is adapted from [93].

The Interstate-24 Video dataset utilizes 3 sets of coordinates utilized throughout I-24 MOTION system [34] datasets:

- **Image Coordinates:** are given in pixels. $(y^{(im)}, x^{(im)})$ denotes the row and column of the specified pixel. By convention the top left pixel is (0,0).
- **State Plane Coordinates:** specify a rectilinear and orthogonal coordinate system. The EPSG 2274 state plane coordinate system for Tennessee is specified in feet relative to a known survey point. $(x^{(st)}, y^{(st)})$ indicates the coordinate (in feet) along the first (roughly east-west) and second (roughly north-south) coordinate axis defined by the state plane coordinate system. (Note that a common conversion from state plane coordinates to latitude/longitude coordinates (e.g. WSG84 or NAD83) can be utilized if desired.) A third orthogonal coordinate axis (z-axis) is defined and corresponds to distance off the roadway, such that $z^{(st)} = 0$ for all points on the roadway plane.
- **Roadway Coordinates:** are defined such that the primary (x) axis lies along the median (or more precisely, midway between the two interior yellow lines for the interstate) at all points within the instrument extents, and the secondary (y) axis is defined locally to perpendicular to the primary axis at all points along the roadway. All coordinates with a distance from the primary axis less than the local radius of curvature (including all points on the roadway) have a unique $(x^{(r)}, y^{(r)})$ coordinate. By left-hand rule convention, we define the positive y-axis to be in the direction of the eastbound roadway lanes at all points along the roadway.

O.1 Notation

Throughout the rest of this Appendix to disambiguate the various coordinate systems, the following notation is used:

- x,y, and z refer to coordinate axes. A superscript (im), (st), or (r) specifies all variables corresponding to a specific coordinate system (e.g. $x^{(st)}$).
- Vectors and matrices in a specified coordinate system are denoted in bold (e.g. $\mathbf{O}^{(st)}$).
- Homography matrices are also listed in bold script, without superscript (e.g **H**).
- A subscript indexes a specific point (e.g. $x^{(st)}_{bbl}$), and subscript i indicates an arbitrary element index from a set of elements (e.g a_i).
- An x,y, or z without a subscript indicates a generic variable along the specified axis within the specified coordinate system.

A list of all variables along with their descriptions is given in Table 15. Transformations between image and state plane coordinates, and transformations between state plane and roadway coordinates are detailed in the next two sections.

O.2 Image $\leftrightarrow$ State Plane Conversion

(parts of this subsection rely on similar definitions and descriptions to the supplement in [95].) A *homography* relates two views of a planar surface. For each camera, we provide homography information such that the 8-corner coordinates of the stored 3D bounding-box annotation can be projected into any camera view for which the vehicle is visible, creating a monocular 3D bounding box within that camera field of view. For each direction of travel in each camera view, for each scene, a homography relating the image pixel coordinates to the state plane coordinate system is defined. (Though the same cameras are used for different scenes, the positions of the cameras changes slightly over time due). A local flat plane assumption is used (the state plane coordinate system is assumed to be piece-wise flat) [344]. A series of correspondence points $a_i = [x^{(im)}, y^{(im)}, x^{(st)}, y^{(st)}, z^{(st)}]$ are used to define this relation, where $(y^{(im)}, x^{(im)})$ is the coordinate of selected correspondence point a in pixel coordinates (row, column) and $(x^{(st)}, y^{(st)}, z^{(st)})$ is the selected correspondence point in state plane coordinates.

All selected points are assumed to lie on the state plane, so $z^{(st)} = 0$ for all selected correspondence points. Visible lane marking lines and other easily recognizable landmarks on the roadway are used as

Symbol	Definition
$\mathbf{H}$	3×3 matrix of homography parameters h_{ij}
$\mathbf{P}$	3×4 matrix of homography parameters p_{ij}
s	homography scale parameter
$x^{(im)}$, $y^{(im)}$	image coordinates (y indicates pixel row and x indicates pixel column)
$x^{(st)}$, $y^{(st)}$, $z^{(st)}$	state plane coordinates
$x^{(r)}$, $y^{(r)}$	roadway coordinates
$\mathbf{O}^{(st)}$	state plane coordinates for object, equal to $[\mathbf{o}_{bbl}^{(st)},\mathbf{o}_{bbr}^{(st)},\mathbf{o}_{btl}^{(st)},\mathbf{o}_{btr}^{(st)},\mathbf{o}_{fbl}^{(st)},\mathbf{o}_{fbr}^{(st)},\mathbf{o}_{ftl}^{(st)},\mathbf{o}_{ftr}^{(st)}]$
$\mathbf{o}_{bbl}^{(st)}$	back bottom left state plane coordinate of object, equal to $[x_{bbl}^{(st)},y_{bbl}^{(st)},z_{bbl}^{(st)}]$
$\mathbf{o}_{c}^{(st)}$	back bottom center state plane coordinate, primary reference coordinate for the object
$\mathbf{o}_{spl}^{(st)}$	state plane coordinates of point on center-line spline ($y^{(r)} = 0$) with the same $x^{(r)}$ coordinate as $\mathbf{o}_{c}^{(st)}$
$\mathbf{O}^{(r)}$	roadway coordinates for object, $[x^{(r)_o}, y^{(r)_o}, l, w, h]$
$x^{(r)}$	generic longitudinal roadway coordinate along curvilinear spline axis
$y^{(r)}$	generic lateral roadway coordinate along axis locally perpendicular to longitudinal roadway coordinate axis
$x_o^{(r)}$	object longitudinal roadway coordinate along curvilinear spline axis
$y_o^{(r)}$	object lateral roadway coordinate along axis locally perpendicular to longitudinal roadway coordinate axis
l,w,h	rectangular prism dimensions (length, width and height)
$F(x^{(r)})$	spline defining state plane coordinate roadway center-line spline parameterized by $x^{(r)}$
$\tilde{G}(x^{(st)})$	spline approximating the center-line spline in roadway coordinates $x^{(r)}$ parameterized by $x^{(st)}$

Table 15: Summary of symbols used in this section.

correspondence points in each camera field of view. Each correspondence point is also labeled in *global information system* (GIS) software, giving the precise GPS / state-plane coordinate system coordinates for each labeled corresponding point. The corresponding pixel coordinates are manually selected in each camera field of view, for each direction of travel on the roadway (see Appendix O.5).

A *perspective transform* (Equation 28) is fit to these correspondence points. We first define a 2D perspective transform which defines a linear mapping (Equation 27) of points from one plane to another that preserves straight lines. The correspondence points are then used to solve for the best perspective transform $\mathbf{H}$ as defined in equation 28, where s is a scale factor.

$$s \begin{bmatrix} x_i^{(st)} \\ y_i^{(st)} \\ 1 \end{bmatrix} \sim \mathbf{H} \begin{bmatrix} x_i^{(im)} \\ y_i^{(im)} \\ 1 \end{bmatrix} \tag{27}$$

where $\mathbf{H}$ is a 3×3 matrix of parameters:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{28}$$

For each camera field of view and each direction of travel, the best perspective transform $\mathbf{H}^*$ is determined by minimizing the sum of squared re-projection errors according to equation 29 as implemented in OpenCV's *find_homography* function [348]:

$$\mathbf{H}^* = \underset{\mathbf{H}}{\arg\min} \sum_i \left(x_i^{(st)} - \frac{h_{11}x_i^{(im)} + h_{12}y_i^{(im)} + h_{13}}{h_{31}x_i^{(im)} + h_{32}y_i^{(im)} + h_{33}} \right)^2 + \left(y_i^{(st)} - \frac{h_{21}x_i^{(im)} + h_{22}y_i^{(im)} + h_{23}}{h_{31}x_i^{(im)} + h_{32}y_i^{(im)} + h_{33}} \right)^2 \tag{29}$$

The resulting matrix $\mathbf{H}^*$ allows any point lying on the plane within the camera field of view to be converted into state plane coordinates. The corresponding matrix $\mathbf{H}_{inv}$ can easily be obtained to project state plane coordinates on the $z = 0$ plane into image coordinates. However, since each vehicle is represented by a 3D bounding box, the top corner coordinates of the box do not lie on the ground plane. A 3D perspective transform $\mathbf{P}$ is needed to linearly map coordinates from 3D state plane coordinate space to 2D image coordinate space, where $\mathbf{P}$ is a 3×4 matrix of parameters:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tag{30}$$

and $\mathbf{P}$ projects a point in 3D space (x', y', z') into the corresponding image point (x, y) according to:

$$\mathbf{P} \begin{bmatrix} x^{(st)} \\ y^{(st)} \\ z^{(st)} \\ 1 \end{bmatrix} \sim s' \begin{bmatrix} x^{(im)} \\ y^{(im)} \\ 1 \end{bmatrix} \tag{31}$$

where s' is a new scaling parameter. By observing the case where $z^{(st)} = 0$, it is evident columns 1,2, and 4 of $\mathbf{P}$ are equivalent to the columns of $\mathbf{H}_{inv}$ and can be fit in the same way. Thus, we need only solve for column 3 of $\mathbf{P}$. Next, we note as in [344] that $(\frac{p_{11}}{p_{31}}, \frac{p_{21}}{p_{31}})$ is the vanishing point (in image coordinates) of perspective lines drawn in the same direction as the state plane coordinate x-axis. The same is true for the 2nd column and the state plane coordinate y-axis, the 3rd column and the state plane coordinate z-axis, and the 4th column and the state plane coordinate origin.

Thus, to fully determine $\mathbf{P}$ it is sufficient to locate the vanishing point of the z-axis in state plane coordinates and to estimate the scaling parameter p_{33}. The vanishing point is located in image coordinates by finding the intersection point between lines drawn in the z-direction. Such lines are obtained by manually annotating vertical lines in each camera field of view. The scale parameter is estimated by minimizing the sum of squared reprojection errors defined in equation 32 for a sufficiently large set of state plane coordinates and corresponding, manually annotated coordinates in image space.

$$\begin{aligned}\mathbf{P}^* = \underset{p_{33}}{\arg\min} \sum_i & \left(x_i^{(im)} - \frac{p_{11}x_i^{(st)} + p_{12}y_i^{(st)} + p_{13}z_i^{(st)} + p_{14}}{p_{31}x_i^{(st)} + p_{32}y_i^{(st)} + p_{33}z_i^{(st)} + p_{34}} \right)^2 + \\ & \left(y_i^{(im)} - \frac{p_{21}x_i^{(st)} + p_{22}y_i^{(st)} + p_{23}z_i^{(st)} + h_{24}}{p_{31}x_i^{(st)} + p_{32}y_i^{(st)} + p_{33}z_i^{(st)} + h_{34}} \right)^2\end{aligned} \tag{32}$$

The resulting 3D perspective transform $\mathbf{P}^*$ allows for the lossless conversion of points in roadway coordinates to the corresponding points in image coordinates. Observing that a lossless conversion from image coordinates to state plane coordinates is available provided that the converted point lies on the $z^{(st)} = 0$ plane, it is possible to precisely convert a rectangular prism from image space to state plane coordinates by i.) converting the footprint of the prism near-losslessly into state plane coordinates (the only source of error comes from a set of 4 image coordinates that cannot be perfectly converted into a rectangle in state plane coordinates), ii.) shifting the footprint in state plane coordinates along the z-axis, iii.) re-projecting the resulting points back into the image, iv.) comparing the reprojected "top points" to the original top of the rectangular prism in image coordinates, and v.) adjusting the height iteratively to minimize the re-projection error until convergence.

O.3 State Plane → Roadway Coordinate System Conversion

Next, we consider the conversion of points in state plane coordinates to roadway coordinates. In most cases, we care to convert a set of state plane coordinate points roughly in a rectangular prism (i.e. vehicle 3D bounding box) into roadway coordinates; thus, we define this conversion for a rectangular prism. A single point can be converted between state plane coordinates and roadway coordinates by treating it as a rectangular prism with zero length, width and height.

Let $\mathbf{O}^{(st)}$ be a 3D bounding box representation in state plane coordinates, an 8×3 matrix of x,y, and z coordinates for each corner of the box. (Note that these corners need not exactly correspond to an orthogonal rectangular prism, but the roadway coordinate equivalent will be exactly orthogonal so some truncation will occur.) We reference, for example, the back bottom right (from the perspective of the rear of the vehicle) of object $\mathbf{O}^{(st)}$ as $\mathbf{o}_{bbr}^{(st)} = [x_{bbr}^{(st)}, y_{bbr}^{(st)}, z_{bbr}^{(st)}]$, such that = $\mathbf{O}^{(st)} = [\mathbf{o}_{bbl}^{(st)}, \mathbf{o}_{bbr}^{(st)}, \mathbf{o}_{btl}^{(st)}, \mathbf{o}_{btr}^{(st)}, \mathbf{o}_{fbl}^{(st)}, \mathbf{o}_{fbr}^{(st)}, \mathbf{o}_{ftl}^{(st)}, \mathbf{o}_{ftr}^{(st)}]$. (For the single-point case described above, all 8 corner coordinates are identical).

Next, Let $\mathbf{O}^{(r)} = [x_o^{(r)}, y_o^{(r)}, l, w, h]$ be the corresponding object representation of $\mathbf{O}^{(st)}$ in roadway coordinates. $x^{(r)}$ and $y^{(r)}$ are the roadway coordinate longitudinal and lateral coordinates (in feet), and l, w,and h are the length, width, and height of the object respectively (in feet).

Let $\mathbf{o}^{(st)_c}$ denote the back bottom center coordinate of object $\mathbf{O}^{(st)}$. By convention, this point is referenced as the primary position of object $\mathbf{O}^{(st)}$. Let $\mathbf{o}^{(st)_{spl}}$ denote the point on the center-line spline (i.e. $y^{(r)} = 0$) with the same $x^{(r)}$ coordinate as $\mathbf{o}_c^{(st)}$.

Let F be the second-order spline parameterizing the roadway center-line in state plane coordinates. In other words, F defines the longitudinal curvilinear axis $y^{(r)} = 0$ along this spline. F is fit by manually labeling a sufficiently large number of points along the interior yellow line for both directions of travel (in state plane coordinates). A spline is fit to each yellow line, and a third spline is fit to lie precisely halfway between these two splines. Spline control points are selected at suitably sparse intervals (200 foot minimum spacing) such that the spline is relatively smooth while still capturing the roadway curvature (see Appendix O.6).

Given $\mathbf{O}^{(st)}$, we first obtain l,w and h by computing the average distance between points on the front and back, left and right, or top and bottom of the vehicle respectively. Next, we obtain $\mathbf{o}_c^{(st)}$ by computing the average $x^{(st)}$ and $y^{(st)}$ state plane coordinates of the 4 back rectangular prism corners.

Next, we solve for $x_o^{(r)}$ by solving the following optimization:

$$x_o^{(r)} = \arg\min_{x^{(r)}} ||(F(x^{(r)}), \mathbf{o}_c^{(st)}||_2 \tag{33}$$

using L2-norm (Euclidean distance) between the two points in state plane coordinate space. In other words, determine the point on the roadway spline closest to the back center of the rectangular prism $\mathbf{o}_c^{(st)}$. This minimizing point is the corresponding roadway longitudinal coordinate $x_o^{(r)}$, and the distance from the minimum distance point is roadway lateral coordinate $y_o^{(r)}$.

$$y_o^{(r)} = \min_{x^{(r)}} ||(F(x^{(r)}), \mathbf{o}_c^{(st)}||_2 \tag{34}$$

Noting that the I-24 MOTION roadway segment has monotonically increasing $x^{(st)}$ coordinate, a secondary spline $\tilde{G}(x^{(st)})$ is defined to parameterize $x^{(r)}$ as a function of $x^{(st)}$, which yields a good initial guess for the closest roadway longitudinal coordinate for a given point in state plane coordinates. This optimization can then be solved to arbitrary precision, yielding the complete roadway coordinate for the object $\mathbf{O}^{(r)} = [x_o^{(r)}, y_o^{(r)}, l, w, h]$.

Constant Yellow Line Constraint: It is observed that points along the yellow line for each roadway direction of travel have non-constant y-position due to the varying width of the median. It is desirable that the yellow line (and by extension each set of lane-dividing markings) has constant y position. We finally apply the following shift:

$$y_o^{(r)} += (C - \gamma(x_o^{(r)})) \tag{35}$$

where C is the desired constant yellow line y-coordinate (in this case -12 for westbound-side coordinates and +12 for eastbound-side coordinates) and $\gamma(x_o^{(r)})$ represents the uncorrected yellow-line coordinate at the given x-position per roadway side. Note that this creates a discontinuity in the coordinate system near $y = 0$ on each side, but these portions of the coordinate system are not used for objects on the roadway.

O.4 Roadway → State Plane Conversion

Reverse Yellow Line Constraint: First, the inverse yellow-line shift must be applied to the y-coordinate:

$$y_o^{(r)} -= (C - \gamma(x_o^{(r)})) \tag{36}$$

Next, given roadway coordinates for an object $\mathbf{O}^{(r)}$, first find the corresponding point on the roadway center-line spline in state plane coordinates $\mathbf{o}_c^{(st)}$ according to:

$$F(x_o^{(r)}) = \mathbf{o}_{spl}^{(st)} \tag{37}$$

To obtain the back center coordinate $\mathbf{o}_c^{(st)}$, we must offset $\mathbf{o}_{spl}^{(st)}$ by length $y^{(r)}$ in the direction perpendicular to the roadway centerline spline at $\mathbf{o}_c^{(st)}$. Let $\overrightarrow{\mathbf{u}}_F$ be the unit vector in the same direction as the derivative spline F', and let $\overrightarrow{\mathbf{u}}_{1/F}$ be the unit vector in the perpendicular direction (along the state plane, i.e. $z^{(st)} = 0$. Note that care should be given to ensure that the positive direction of $\overrightarrow{\mathbf{u}}_{1/F}$ points towards the eastbound side of the roadway with positive $y^{(r)}$.) Then, $\mathbf{o}_c^{(st)}$ is given by:

$$\mathbf{o}_c^{(st)} = \mathbf{o}_{spl}^{(st)} + y^{(r)} \cdot \overrightarrow{\mathbf{u}}_{1/F} \tag{38}$$

From here, the corner state plane coordinates for the right and left coordinates of the rectangular prism can be obtained by offsetting $\mathbf{o}_c^{(st)}$ by $\pm\frac{1}{2}$ times w in the direction of $\overrightarrow{\mathbf{u}}_{1/F}$, and the front coordinates of the rectangular prism can similarly be obtained by offsetting by l in the direction of $\overrightarrow{\mathbf{u}}_F$ or in the opposite direction for objects on the westbound or negative $y^{(r)}$ side of the roadway. Similarly, the top coordinates can be obtained by offsetting by a factor of h in the $z^{(st)}$ direction. The direction of travel for an object can be obtained as the sign of the $y^{(r)}$ coordinate (negative for WB, positive for EB).For example, for an eastbound object the front top left coordinate can be obtained as:

$$\mathbf{o}_{ftl}^{(st)} = \mathbf{o}_c^{(st)} - \frac{1}{2} \cdot w \cdot \overrightarrow{\mathbf{u}}_{1/F} + l \cdot \overrightarrow{\mathbf{u}}_F + h \cdot [\mathbf{0}, \mathbf{0}, \mathbf{1}] \tag{39}$$

O.5 Correspondence Point Labeling

A brief description and a few pictures of overhead and camera fields of view.

Correspondence points are obtained for visible lane marking lines and other easily recognizable landmarks on the roadway are used as correspondence points in each camera field of view. Labeling is carried out in a custom OpenCV GUI, with semi-automated point selection to speed this process over the 230+ cameras utilized in this work. An average frame from each camera is used to minimize the presence of occluding vehicles during labeling. Figure O.13 shows an example of labeled points and the labeled relevant field of view for a single camera. The corresponding pixel coordinates for each manually selected point in each camera field of view, are stored, for each direction of travel on the roadway, each with a unique identifier. Typically, at least 10 roadway markings or 40 correspondence points are labeled per roadway direction of travel. Additionally, note that this initial correspondence point selection was performed for a day a few weeks prior to the day on which video is recorded in this work.

Each correspondence point is also labeled in *global information system* (GIS) software, giving the precise GPS / state-plane coordinate system coordinates for each labeled corresponding point. The same unique identifier is assigned to each corresponding point previously labeled within a camera image. Additionally, points along each yellow line were labeled periodically (every ∼50 feet). Figure O.15 shows an example of labeled points in the state plane aerial imagery.

O.6 Centerline Spline Fitting

The following procedure was used to fit the center-line spline for the roadway coordinate system ($y^{(r)} = 0$):

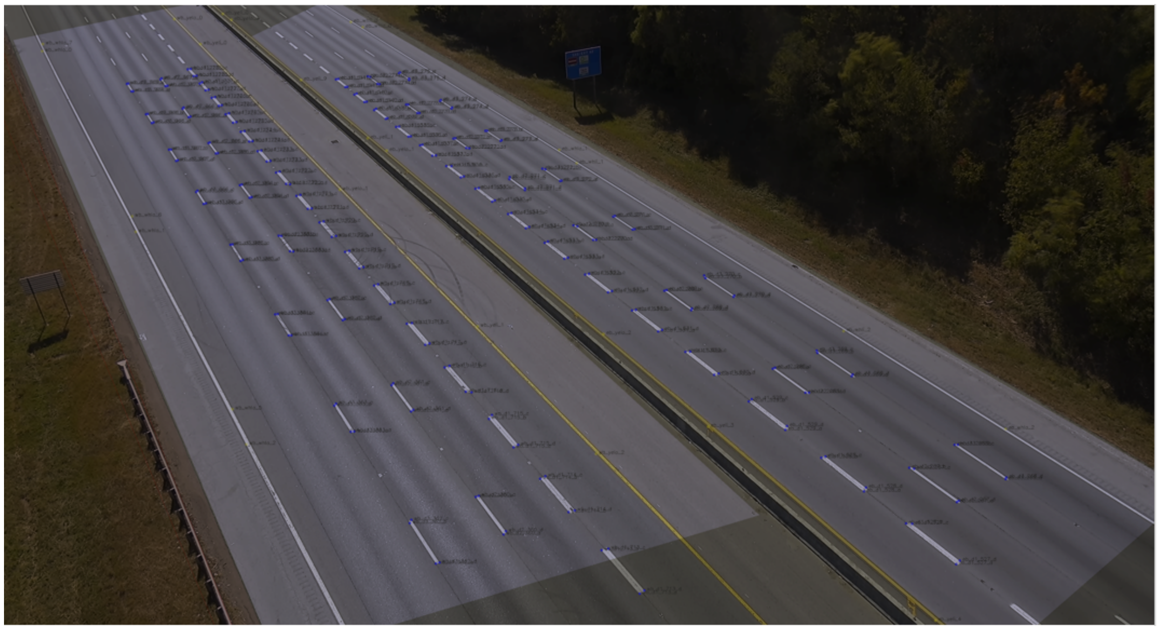


Figure O.13: Correspondence points (blue) and fields of view (shaded polygons) for each roadway direction of travel, labeled in a single camera field of view (P27C01). Around 100 correspondence points are visible for each side of the roadway.

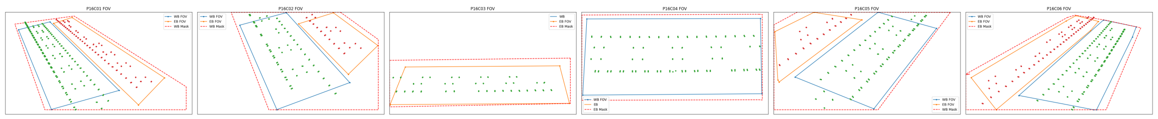


Figure O.14: Correspondence points (green for WB red for EB) labeled in camera imagery for all cameras on one pole (P17). Fields of view (blue for WB and orange for EB) and a mask denoting relevant portions of the image for tracking (dotted red polygon) are also shown. Similar plots are included in a separate file for all cameras. Credit: Gergely Zachar.

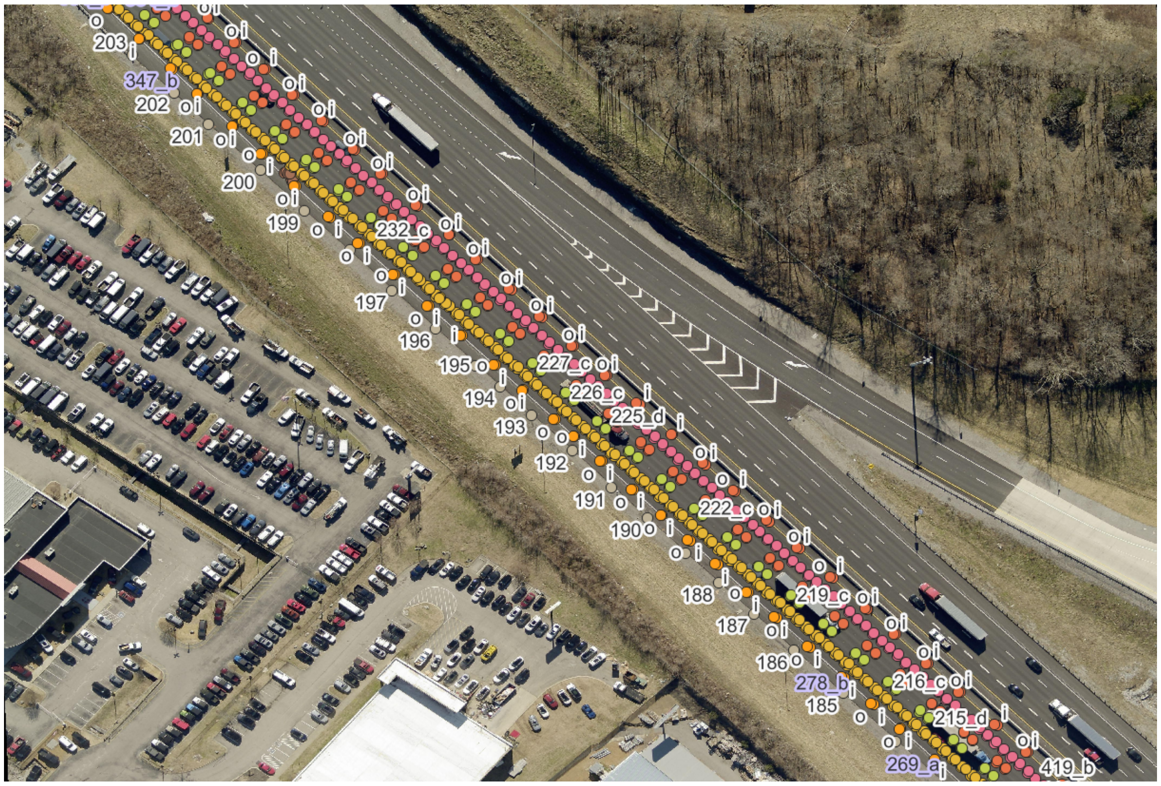


Figure O.15: Correspondence points labeled in aerial imagery using the GIS tool for a very small subset (∼500 feet) of roadway, for one direction of travel only.

1. For each roadway direction of travel, fit a spline parameterizing the $(x^{(st)}, y^{(st)})$ points as a function of u. Let $F_{(WB)}(u)$ and $F_{(EB)}(u)$ denote these splines. All splines implemented in this work are based on SciPy's spline package [368] and are constrained to have a minimum control point spacing of 200 feet to ensure smoothness.

2. Moving along the EB yellow line spline $F_{(EB)}(u)$, sample points at fine (1 foot) intervals. For each sampled point u, find the closest point on $F_{(WB)}(u)$ u'. Store the midpoint $u*$.

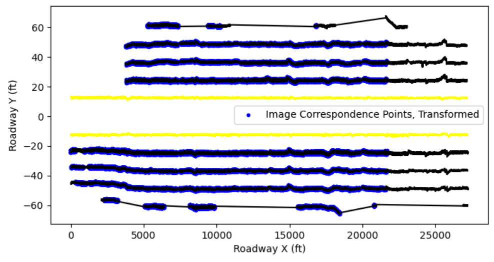
3. Fit another spline $F_{med}(u)$ to the set of midpoints from the previous step.

4. Sample $F_{med}(u)$ at fine (0.1 foot intervals). Compute the distance between each consecutive pair, and compute the cumulative distance along the spline to each point via finite difference approximation.

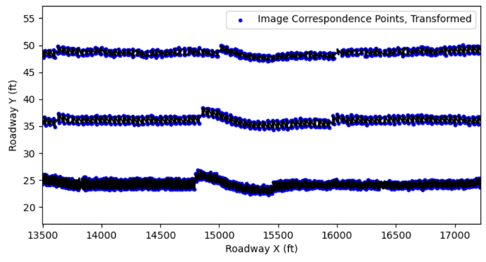
5. Re-parameterize $F_{med}(u)$ such that each point on the midpoint corresponds to the integrated distance along the spline to that point. This yields the final spline $F(x^{(r)})$ used above.

Lastly, $\gamma(x^{(r)}))$ must be computed for each roadway direction of travel. This is done by sampling $F(x^{(r)})$ for each direction at regular (5-foot) intervals and recording the distance to the closest point on each of $F_{(WB)}$ and $F_{(EB)}$. Since these offsets change very little ($\sim$1 foot per mile), the offsets are stored in a lookup table rather than fitting a true offset function. Thus, $\gamma(x^{(r)}))$ returns the recorded offset from $F(x^{(r')})$ to $F_{(WB)}$ or $F_{(EB)}$ for the closest sampled value $x^{(r')}$.

Figure O.16a shows the labeled points from aerial imagery (yellow and black lines) and the points labeled in each image (blue dots), transformed into roadway coordinates. Figure O.16b shows the largest magnitude shift in labeled correspondence points, when converted to the roadway coordinate system. Such "discontinuities" are due to slight misalignment between aerial images taken during different passes of the photographing aircraft, and result in small (less than 2 foot) errors in all cases. While these misalignments could be corrected for by applying a smoothing to the correspondence points labeled within the aerial imagery, this was ultimately decided against because the smoothing would need to be performed with respect to, essentially, the roadway coordinate system spline itself, which is in turn a product of the labeled aerial imagery points. This would produce complex and less well-understood artifacts in the resulting final coordinate system.



(a)



(b)

Figure O.16: (a) correspondence points labeled in aerial imagery (yellow and black) and in individual camera fields of view (blue) are projected into the roadway coordinate system. (b) A close-up detailing a type of artifact visible in the coordinate system as a result of misalignment in the aerial imagery.

P Homography Re-estimation Methods for the I-24 Video Dataset (Ch. 7)

In ideal circumstances homographies can be estimated once (as discussed in the previous section), and used continuously until some drastic change in the system, (e.g. the roadway section is rebuilt or the lane

Appendix P is adapted from [93].

markings are repainted). Aside from these rare events there are several other factors which significantly degrade accuracy; long term events can be the replacement of the camera, inaccuracies in the pan-tilt mechanism during homing, settlement of the foundation or the seasonal temperature change. Although these can be dealt with occasional re-calibration, the *sunflower effect* requires constant re-estimation of the homographies. This effect, the tilting of the metal infrastructure poles due to the differential heating of the sun and shade-facing sides, can cause significant homography errors (sometimes greater than 10 feet, see Figure P.17 for a typical camera example), both in timescales of hours for the daily warm up and cool down cycle, and also in the timescales of minutes, caused by the varying cloud coverage (see Figure P.18, showing how the positional drift in homographies varies over just 12.5 minutes). A video is included in supplementary material showing the magnitude of drift over the course of a day. The camera movement can easily be seen by viewing the relation between the lane markers and the ROI rectangle (which maintains constant pixel coordinates throughout the video). For reference, a typical dash line is 10 feet long.

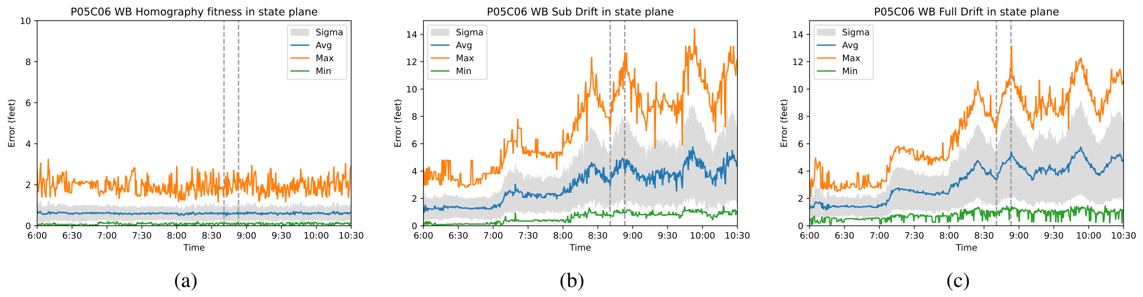


Figure P.17: Homography goodness of fit and uncorrected drift between reference homography and true scene homography according to two metrics (Sub Drift and Full Drift) for a single typical camera. Horizontal grey dashes show the two time instances for Figure P.18.(Credit: Gergely Zachár).

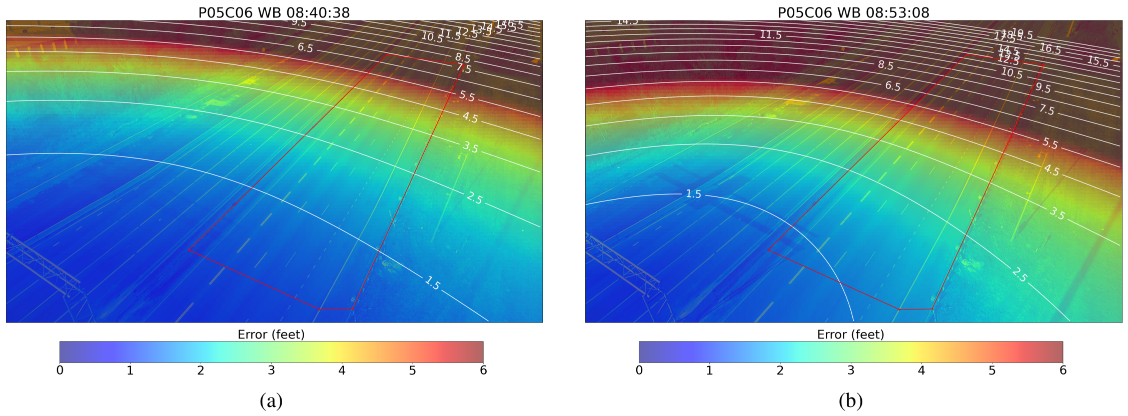


Figure P.18: Full drift for the same camera at two time instances. For a given error value X, this indicates that the original reference homography and the true homography for the scene map the same image coordinate to points in the state plane X feet apart. The westbound field of view (i.e. the most important portion of the image) is shown as a red polygon. Credit: Gergely Zachar.

Note, that tilting not only degrades single-camera accuracy, but due to the multiple pole architecture and the camera FOVs, homography errors cause significant misalignment in state plane coordinates between poles, resulting in vehicle tracking fragmentations . (Cameras on the same pole are less susceptible because they are rigidly mounted and roughly ”moving” together.) Thus, the re-estimation of the homography is necessary on relatively short timescales, or at a minimum on a daily basis.

Since manual re-annotation is not possible for continuous operation, a feature point re-identification method is developed, along with re-estimation methods for a static daily and for a dynamic, time-dependent homography.

All methods and filtering steps are based on the assumption that the image is only slightly distorted from the manually labeled ground truth. This assumption is necessary, because the homography and the re-discovery relies on the lane markings which are non-unique and repetitive, thus a significant shift can cause a mis-identification and misalignment. All presented methods are *offline* (they utilize both past and future information relative to the specified time instance (i.e. use a non-casual filter)).

Automated homography generation is divided into two distinct steps. First the feature points are re-discovered and a new homography is calculated for a single time instance. Second, a refinement step operates on the homographies themselves of which removes outliers, filters, and aggregates over time. This approach provides robustness for the final estimates based on the assumption that the re-estimation provides good results *in general* but are prone to outliers and errors. To comply to this idea the implemented refinement processes are draconian, and tuned to minimize false positives at the expense of occasionally removing some true positives. The "good in general" assumption is empirically tested, by rigorously inspecting the results and fine tuning the filter parameters of each steps. In case of faulty estimates manual inspection always shows problems which could cause confusion even for human observers or significant noise and drop in image quality (e.g.: at dawn).

P.1 Notation

The following notation convention is used throughout the remainder of this Section. Also see Table 16 for a list of utilized symbols and their meanings.

- Mathcal script (e.g. $\mathcal{I}$) is used for sets of points. Bold Text (e.g. $\mathbf{H}$) is used for vectors and matrices.
- The subscript t denotes a set of points or a homography for a specific time instance (e.g. $\mathcal{I}_t$)
- No subscript denotes the initial reference points or homography (e.g. $\mathcal{I}$, $\mathcal{A}$, and $\mathbf{H}$)
- The prime symbol denotes rediscovered points (e.g. $\mathcal{I}'_t$, which corresponds to subset $\mathcal{A}_t$) , or homographies based on rediscovered points (e.g.: $\mathbf{H}'_t$)
- a superscript ^{s} denotes an image-to-image homography based on SIFT-FLANN matching (e.g. $\mathbf{H}^s_t$)
- A superscript asterisk symbol $*$ denotes a homography *estimate* calculated with methods presented in the previous Appendix (e.g.: $\mathbf{H}^*$).
- An arrow $\mathcal{I} \xrightarrow{\mathbf{H}}$ indicates a linear transformation on the point set $\mathcal{I}$ according to homography $\mathbf{H}$. (e.g. $\mathcal{I}'_t \xrightarrow{\mathbf{H}'_t} \hat{=} \mathcal{A}_t$ denotes that the set $\mathcal{I}'_t$ of rediscovered points at time t, projected from image coordinates to state plane coordinates by $\mathbf{H}'_t$, estimates the set of corresponding points labeled in state plane coordinates $\mathcal{A}_t$).

P.2 Homography Generation for a Single Time Instance

This section briefly describes the steps necessary for the homography generation for a single time instance t.

1. Generate a background extracted image to remove the vehicles from the scene. This is implemented as a 1 minute long averaging of frames, with 50% overlap in time. Note that in case of heavy traffic (essentially for stopped vehicles) this time period might not be sufficient for complete removal.

2. Compute an initial alignment estimate between the frame and the reference frame, on which the original annotations were made. This step utilize the SIFT algorithm to find feature points and a FLANN based matcher (both implemented in OpenCV). Note that the detected features can be anywhere on the image, not necessarily corresponding to semantically meaningful points (e.g. grass, trees, parking cars outside the road), and most importantly generally not lie on the plane of the roadway.

3. Based on the SIFT-FLANN generated corresponding points generate an image-to-image homography ($\mathbf{H}^s_t$).

4. Examine $\mathbf{H}^s_t$: transformation (e.g.: translation, rotation, scale) should be minimal, based on the slight distortion assumption. Otherwise discard the current time instant, because the calculated alignment is likely from erroneously matched points. No further steps are possible.

Symbol	Definition
$\mathcal{I}$	labeled image points (in image coordinates, unit: pixels)
$\mathcal{I}_t$	subset of the labeled image points at time t, corresponding to the successfully rediscovered points ($\mathcal{I}_t \subset \mathcal{I}$)
$\mathcal{I}'_t$	rediscovered image points ($\lvert\mathcal{I}'_t\rvert \leq \lvert\mathcal{I}\rvert$)
$\mathcal{I}^s_t$	labeled image points after perspective transformation, utilizing the SIFT-FLANN matcher, such that $\mathcal{I} \xrightarrow{\mathbf{H}^s_t} \mathcal{I}^s_t$
$\mathcal{I}^*_t$	”labeled” image points generated such as $A \xrightarrow{\mathbf{H}^{*\neg}_t} \mathcal{I}^*_t$
$\mathcal{A}$	aerial points (in state plane coordinates, unit: feet)
$\mathcal{A}_t$	subset of the aerial points at time t, corresponding to the successfuly rediscovered points ($\mathcal{A}_t \subset \mathcal{A}$)
$\mathbf{H}$	homography such that $\mathcal{I} \xrightarrow{\mathbf{H}} \mathcal{A}$
$\mathbf{H}^{\neg}$	inverse homography such that $\mathcal{A} \xrightarrow{\mathbf{H}^{\neg}} \mathcal{I}$
$\mathbf{H}'_t$	homography such that $\mathcal{I}'_t \xrightarrow{\mathbf{H}'_t} \mathcal{A}_t$
$\mathbf{H}^*$	new *static homography estimate* by method from Appendix P.3
$\mathbf{H}^*_t$	new *dynamic homography estimate* for time t by methods from Sections P.4-P.5
$\mathbf{H}^s_t$	projection (image to image) generated with the SIFT-FLANN matcher results

Table 16: Notations for various point sets and homographies utilized in this work.

5. Binarize the averaged frame via OpenCV’s *threshold* function [348] and run a contour detection algorithm. This provides good results because lane markings are high contrast, distinct features.
6. Transform the originally labeled feature points $\mathcal{I}$, i.e. the four corner points of the lane markers, with the calculated $\mathbf{H}^s_t$ (producing $\mathcal{I}^s_t$) and calculate the geometric center for each marker. This provides a seed point for rediscovery.
7. Select contour areas which include a seed point, thus creating a set of candidate for lane marks.
8. For each marker, select the 4 corner points on the corresponding contour that produce the largest area, thus creating a quadrilateral.
9. Filter the candidates by contour area, contour area to quadrilateral area ratio, ratio of area compared to the labeled reference area. This step is important to remove candidates which are merged, partially occluded, or otherwise not rediscovered correctly.
10. Select the proper labels and store the 4 corner points as feature points of each remaining markers. With this step the rediscovery steps are complete, yielding $\mathcal{I}'_t$.
11. Run a homography matrix estimator (which utilizes RANSAC) between the rediscovered $\mathcal{I}'_t$ and the corresponding aerial $\mathcal{A}_t$ points. Remove the feature points which are considered outliers by the algorithm. In the current implementation these are points of which are further than $\tilde{2}$ feet on the state plane. The resulting homography for time instance t is written as $\mathbf{H}'_t$.

The result of the listed process is a set of rediscovered feature points ($\mathcal{I}'_t$) and a new homography ($\mathbf{H}'_t$) for a given time instant t. Note that although the steps involve heavy filtering and outlier detection the resulting, a standalone homography should not be used without further processing for the following reasons:

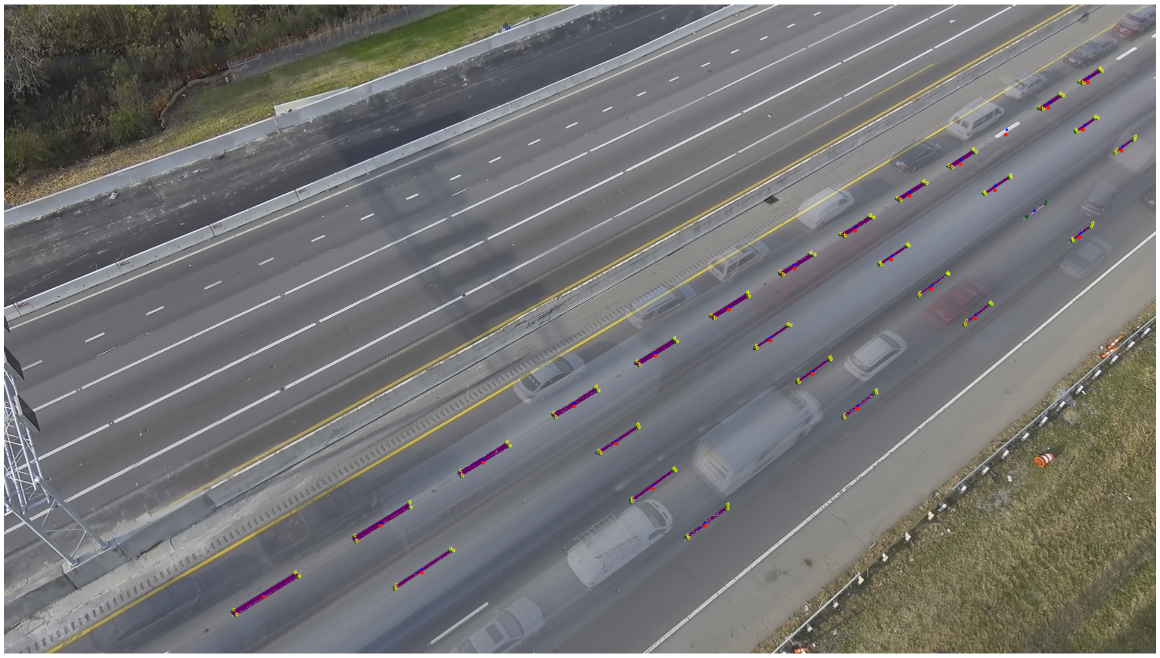


Figure P.19: An example of SIFT-FLANN based seed points from step 6 (red dots), detected contours (purple outlines) and re-detected dash corner points (yellow dots) for a 1-minute average frame. Because traffic at this time instance was partially stopped, some vehicle artifacts are visible in the averaged frame. A few dashes are not successfully rediscovered. Credit: Gergely Zachar.

- The number and positions of the rediscovered lane markings are not guaranteed, thus it is possible to only rediscover lane markings which lie on the same line, thus providing poor estimates for the perpendicular axis.
- Lane markings ”far” from the camera are hard to properly detect due to their small area and slight camera movements can cause misidentification.
- The RANSAC based homography estimator does not guarantee that the selected points are proper. In extreme cases it can select an erroneous subset and fit a good homography to it. As discussed previously, the lane markings are non-unique and repetitive: usually there are multiple subsets of points with nearly identical relative arrangement.

P.3 Static Homography Generation

To counter for the long term homography errors and somewhat compensate for the sunflower effect a single homography can be generated for a given time interval, e.g.: the length of the recording or for the desired time range. This method has the advantage of time-invariance (simplicity) and can utilize all the instantaneous estimates over the specified window, resulting in ample redundancy for outlier removal. The method includes two steps:

1. In the preparation phase the outliers are removed, i.e. the homography estimates of which significantly differ from the others. The current implementations considers a homography an outlier if any of its component (in the 3x3 matrix) deviates more than 30% from the arithmetic mean of all estimates. Note that this process is iterative.
2. The main step is the averaging, i.e. calculating the arithmetic mean for each matrix component to produce $\mathbf{H}^*$.

P.4 Dynamic Homography Generation

To counter for both long and short term homography problems a dynamic, time dependent method is proposed. This solution is capable of modeling short-term camera motion (e.g. from the sunflower effect), but is less trivial implement and use compared to the static version because it requires modifying the homography as a function of time.

The underlying idea of this method is the ”smoothing” of the original estimates, based on the temporally nearby values. Note that this is not straightforward because estimates are not necessarily available at all time instances, e.g.: in case of heavy traffic the occlusion prevents homography estimation, similarly at the beginning and at the end of the recording there is no preceding and subsequent information available. (This is especially problematic at dawn and dusk where the even the recorded images are noisy and blurred thus making feature rediscovery hard or even impossible.) Thus a method is necessary of which is capable to adapt to these situations. An important and necessary property of a good dynamic homography is that it provides ”smooth” variability over time, preventing discontinuities in positional data (which could in turn cause fragmentations during object tracking.)

The proposed solution includes three steps:

1. In the preparation phase the outliers are removed. This step is exactly the same as in the static case.
2. A ”window size” parameter is computed, based on the available number of homography estimates in a specified temporal neighbourhood around the given time instance. This step ensures that in case of missing estimates the window size is larger, thus accommodating more data points.
3. The final step is the ”smoothing” of each homography matrix component, where a Gaussian kernel function is used over time. The shape of the function, or more precisely the variance is determined by the previous step.

Note that this method can provide a homography estimate $\mathbf{H}_t^*$ at any time instance, not just when instantaneous estimates are available. In our implementation a new estimate is generated at 10 second intervals.

P.5 SIFT-FLANN-based homography generation (existing)

No lane marking rediscovery is performed. Instead, the set of shifted points ($\mathcal{I}_t^s$) generated from the original labels ($\mathcal{I}$) is direcly used from Appendix P.2 step 6 and above to fit a homography estimate $\mathbf{H}_t^*$ for time t i.e. $\mathcal{I} \xrightarrow{\mathbf{H}_t^s} \mathcal{I}_t^s \xrightarrow{\mathbf{H}_t^*} \mathcal{A}$.

Q Homography Error Metrics for the I24-Video Dataset (Ch. 7)

Next, we turn to defining metrics for assessing the effectiveness and accuracy of the proposed point rediscovery and homography re-estimation metrics. This Appendix provides an overview of the possible metrics and their properties.

Q.1 Notation

In addition to the notation defined in the previous section, the comparison operator E is defined as a function of two sets of points ($\mathcal{B},\mathcal{C} \subset \mathbb{R}^2$). The operator calculates the point-wise L2-norm between the common subset of points shared by $\mathcal{B}$ and $\mathcal{C}$, (i.e. the distances between two points which correspond to the same feature), and produces a set of values. The resulting set can be used to calculate aggregate metrics, such as mean, maximum, and standard deviation of error.

$$E(\mathcal{B},\mathcal{C}) = \{||b_i,c_i||_2,...,||b_n,c_n||_2\}\,\forall i \text{ s.t. } b_i \in \mathcal{B}, c_i \in \mathcal{C}, b_i \mathrel{\hat{=}} c_i \tag{40}$$

Q.2 Reference Metrics

Metrics listed in this Appendix are computed by comparing the original reference labels ($\mathcal{I}$ and $\mathcal{I}$) to rediscovered points ($\mathcal{I}'_t$). In some cases the same set of points is transformed with the original reference homography ($\mathbf{H}$ or $\mathbf{H}^{\neg}$) and the instantaneous homography estimate $\mathbf{H}'_t$. They measure i.) the quality of the rediscovery and instantaneous homography re-estimation process, and ii.) the movement of the cameras.

Most metrics presented here can be calculated in both image and state plane coordinate systems; thus we present them in pairs. The resulting error values have units of pixels or distance (feet), respectively. In practical evaluations we utilize the state plane distance (units of feet) as it is invariant to the image resolution and FOV, thus comparable across cameras. As in Appendix O, we use a superscript (im) or (st) to denote the coordinate system for each metric.

- **Sub Drift I:** This metric compares a subset of the originally labeled *image* points to the rediscovered points. The resulting distances can be interpreted as the *drift* caused by the camera movement. Note that in case of perfect alignment the maximum error is zero.

$$\text{SubDriftI}^{(im)} = E(\mathcal{I}_t,\mathcal{I}'_t) \tag{41}$$

$$\boxed{\text{SubDriftI}^{(st)} = E(\mathcal{I}_t \xrightarrow{\mathbf{H}},\mathcal{I}'_t \xrightarrow{\mathbf{H}})} \tag{42}$$

- **Sub Drift II (not used):** This metric compares a subset of the originally labeled *aerial* points to the rediscovered points. The resulting distances are a combined error of the drift caused by the camera movement *and* the homography fitness error. Thus even in case of no movement the error is non-zero, as the homography is not a perfectly fit mapping of $\mathcal{I}$ to $\mathcal{A}$.

$$\text{SubDriftII}^{(im)} = E(\mathcal{A}'_t \xrightarrow{\mathbf{H}^{\neg}},\mathcal{I}'_t) \tag{43}$$

$$\text{SubDriftII}^{(st)} = E(\mathcal{I}'_t \xrightarrow{\mathbf{H}},\mathcal{A}_t) \tag{44}$$

- **Full Drift:** This metric compares the full set of the originally labeled image or aerial points transformed by the reference and the re-estimated homography. The resulting distances can be interpreted as *drift* caused by the camera movement. In the case of no camera movement, the error is zero (except for possible stochasticity in the optimal homography H'_t selected by the RANSAC algorithm utilized.)

$$\text{FullDrift}^{(im)} = E(\mathcal{A} \xrightarrow{\mathbf{H}^{\neg}},\mathcal{A} \xrightarrow{\mathbf{H}'^{\neg}_t}) \tag{45}$$

$$\boxed{\text{FullDrift}^{(st)} = E(\mathcal{I} \xrightarrow{\mathbf{H}},\mathcal{I} \xrightarrow{\mathbf{H}'_t})} \tag{46}$$

Appendix Q is adapted from [93].

- **Sub Fitness:** This metric shows the fitness of the estimated homography, utilizing the rediscovered points and a matching subset of aerial points. This error is caused by a combination of e.g.: inaccurate feature point rediscovery, lens distortion, non flat roadway.

$$\text{Fitness}^{(im)} = E(\mathcal{A}_t \xrightarrow{\mathbf{H}_t'^{\neg}}, \mathcal{I}_t') \tag{47}$$

$$\boxed{\text{Fitness}^{(st)} = E(\mathcal{I}_t' \xrightarrow{\mathbf{H}_t'}, \mathcal{A}_t)} \tag{48}$$

Q.3 Homography Re-estimation Metrics

Metrics listed here are derived from the metrics introduced at the previous section. Substituting the reference labels and homographies with the re-estimated homographies and points sets derived from them. The interpretation and purpose of these metrics are to qualify how good are the new estimations; i.e. a perfect homography provides a perfect alignment to the detections.

- **Sub Drift:** This metric compares a subset of points generated with the re-estimated homography (from the originally labeled points) to the rediscovered points. The resulting distances can be interpreted as a combined error of the re-estimation and the homography fitness error. Therefore even in case of a perfectly aligned homography the error is non-zero. Note that, because the new subset of points are generated from the reference labeled points this metric essentially merge Sub Drift I and II presented in the previous section.

$$\text{SubDrift}^{(im)} = E(\mathcal{I}_t^*, \mathcal{I}_t') \Longrightarrow E(\mathcal{A}_t \xrightarrow{\mathbf{H}_t^{*\neg}}, \mathcal{I}_t') \tag{49}$$

$$\text{SubDrift}^{(st)} = E(\mathcal{I}_t^* \xrightarrow{\mathbf{H}_t^*}, \mathcal{I}_t' \xrightarrow{\mathbf{H}_t^*}) \Longrightarrow E(\mathcal{A} \xrightarrow{\mathbf{H}_t^{*\neg}} \xrightarrow{\mathbf{H}_t^*}, \mathcal{I}_t' \xrightarrow{\mathbf{H}_t^*}) \Longrightarrow E(\mathcal{A}_t, \mathcal{I}_t' \xrightarrow{\mathbf{H}_t^*}) \tag{50}$$

- **Full Drift:** This metric compares the full set of the originally labeled image or aerial points transformed by the the re-estimated and by the instantaneous homography. In effect, these original labeled points are chosen as "proxy points" for the real locations of the correspondence points in the image as they are a full set known to lie near the true locations. The resulting distances can be interpreted as the error of the homography re-estimation process. Note that in case of a perfect alignment the error is zero.

$$\text{FullDrift}^{(im)} = E(\mathcal{A} \xrightarrow{\mathbf{H}_t^{*\neg}}, \mathcal{A} \xrightarrow{\mathbf{H}_t'^{\neg}}) \tag{51}$$

$$\boxed{\text{FullDrift}^{(st)} = E(\mathcal{I} \xrightarrow{\mathbf{H}_t^*}, \mathcal{I} \xrightarrow{\mathbf{H}_t'})} \tag{52}$$

- **Fitness:** Since the fitness metrics does not depend on the re-estimated homography, they are equivalent to the ones discussed in the previous section.

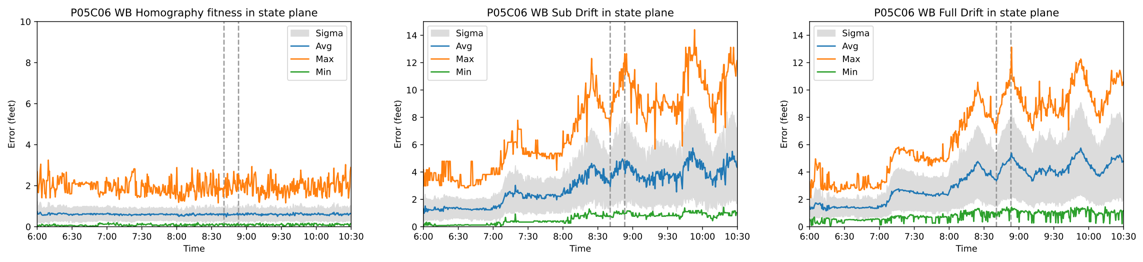
The homography re-estimation performance can be measured by the Full and Sub Drift metrics, but both have caveats to consider: Sub Drift is a better measure of the *real magnitude of the error*, because it incorporates inaccuracies of both the homography and the homography fitness, e.g. errors caused by intrinsic camera problems and the non-flat roadway. On the other hand the Sub Drift metric only includes feature points which are re-detected, thus hard to detect points further from the pole are often missing from the set. This is crucial because the perspective those "far" points account for larger state plane errors than the more easily detectable closer ones. The combined effect of these that the Sub Drift is better in the assessment of the expected minimal error, this is not the case for the maximum. Also worth noting is that Sub Drift does not explictly rely on H_t' being well-fit, so produces a reliable estimate of error even when the fitting of H_t' fails (though in practice this usually indicates poor point rediscovery).

The Full Drift metric is useful to assess the performance of the *homography re-estimation* itself, because in case of perfect alignment the error is reduced to zero. In opposite to the Sub Drift this, metric includes all labeled points, some of which are actually outside the processed ROI, thus resulting in an apparently larger maximum error than during vehicle detection.

Based on these considerations, we select Full Drift (Equation 46) as our primary metric and use it to report the *error* for each method utilized as equation 2 the main text. We also utilize Sub Fitness (Equation 48) as a baseline measure for the error floor of each fit homography ($fitness$ in equation (1) of the main text).

Q.4 Additional Homography Metric Figures

Figure Q.20 shows the different state-plane error metrics for a camera with a long field of view. P05C06 is chosen as it clearly illustrates both long and short-term drift in the homography, as well as the relations between the different metrics: On Figure Q.20a the homography Fittness is shown, representing the minimum achievable accuracy for the vehicle detection. On Figure Q.20b the SubDriftI error is slightly higher than FullDrift shown on Figure Q.20c, because it contains both the fitness and homography error. In this case, the maximum SubDriftI values are higher than the maximum FullDrift values because the rediscovery rate for the feature points is ∼80% for this particular case (relatively high), thus distant points are also likely included. If the drift were more extreme, the most distant correspondence points might not be successfully rediscovered, and the overall reported SubDriftI error may be driven down as a result. This dependence on the number of rediscovered points is one reason why FullDrift is preferred to SubDrift in this work for comparing homography re-estimation performance.



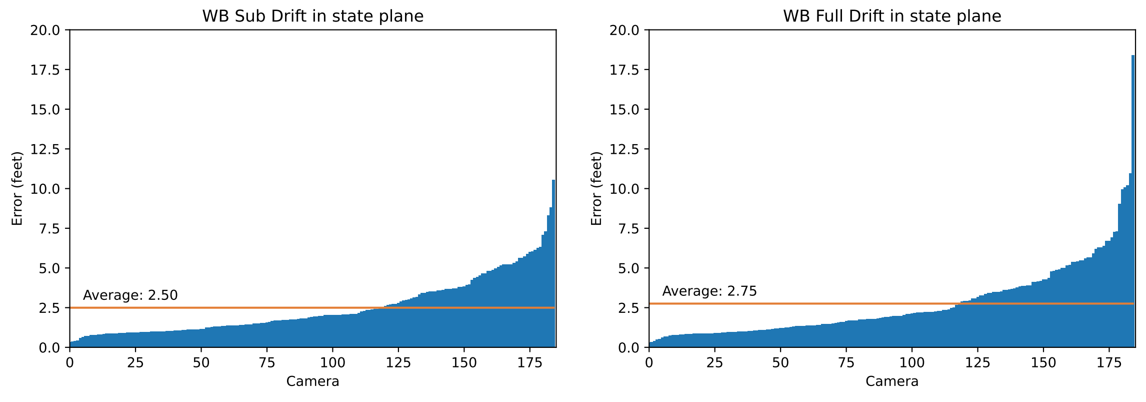
(a) (b) (c)

Figure Q.20: (Repeated from Section 7.2.2.1.) Goodness of fit and uncorrected drift between reference homography and true scene homography according to two metrics (Sub Drift and Full Drift) for a single typical camera. Horizontal grey dashes show the two time instances for Figure P.18. Credit: Gergely Zachar.

Figure Q.21, Q.22 shows the mean FullDrift and SubDrift calculated and averaged over the time of the recording, for all cameras in the system, sorted by magnitude, for each direction of roadway travel. Note that not all cameras have fields of view defined for both sides of the roadway (hence there are more total westbound fields of view than eastbound fields of view). The mean SubDrift is Lower than the mean FullDrift over all cameras, and the maximum mean SubDrift (over all correspondence points and times) is lower than for FullDrift. However as seen in Figure Q.20a, for some time instances the maximum SubDrift can be higher than the maximum FullDrift.

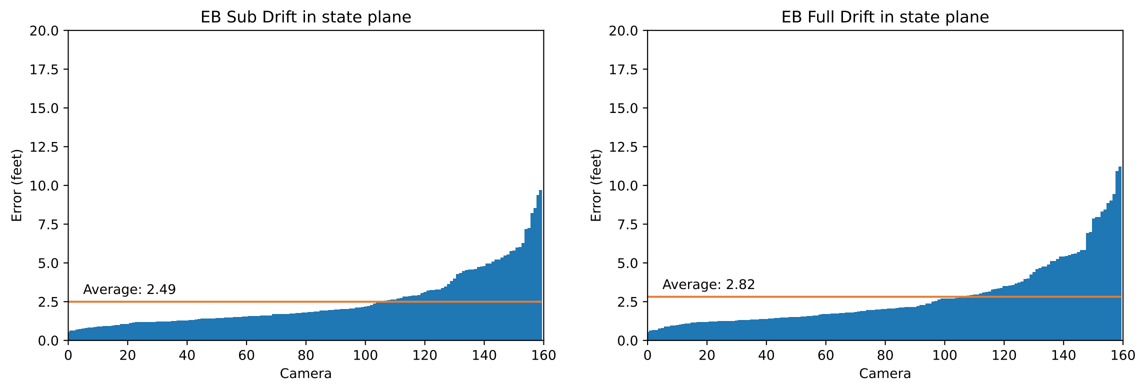
The effectiveness of the homography re-estimation methods is illustrated on Figure Q.23, which shows the additional error above the error floor, utilizing the FullDrift metric, for two cameras. The reference (blue) indicates the resulting error without any mitigation, showing both long term (high mean) and short term (high variance) error. The SIFT-FLANN method (orange) illustrates the performance of an optical flow based ”camera stabilization” solution. The static, all-day average method (green) removes the long term error, although it is mostly unable to remove the error caused by the sunflower effect. Finally since the dynamic homography (red) utilizes nearby (temporal) homography estimations for a given time instance, it can cope with short-term fluctuations. Note that in most cases the all-day average is superior to the SIFT-FLANN method, although there can be time instances where the former can produce better results. Figure Q.23 also illustrates, that long field of view cameras (e.g. P05C06) are more sensitive to camera movement, compared to mainly downwards-facing cameras (e.g. P05C04), thus the error caused by drift is more pronounced for them.

Lastly, figure Q.24 compares the remaining mean average SubDrift and FullDrift errors for each camera after (black) SIFT-FLANN feature-matching, (orange) one-day best fit homography re-estimation, and (red) dynamic homography re-estimation methods relative to original reference homography baseline (blue). The



(a) (b)

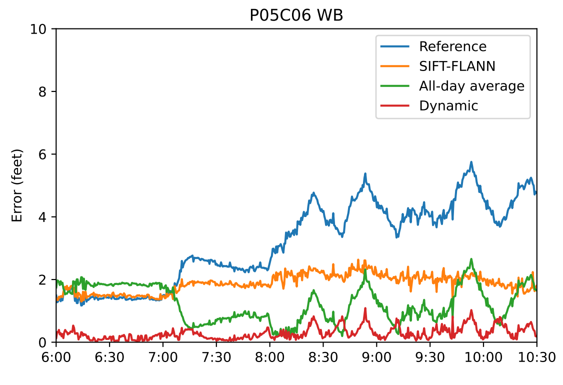
Figure Q.21: Mean averaged SubDrift and FullDrift calculated for all westbound cameras in the system, sorted by error magnitude. Credit: Gergely Zachar.



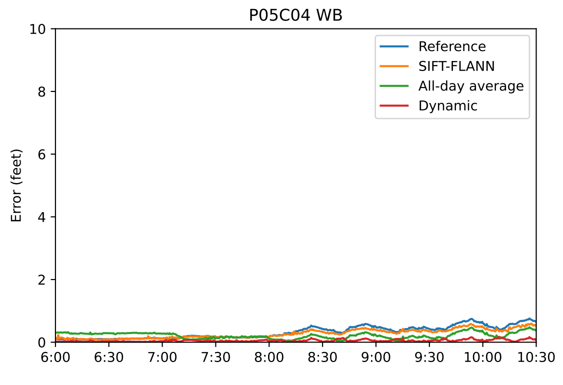
(a) (b)

Figure Q.22: Mean averaged SubDrift and FullDrift calculated for all eastbound cameras in the system, sorted by error magnitude. Credit: Gergely Zachar.

overall trends and relative performance amongst methods is unchanged; in most cases the SIFT-FLANN baseline outperforms the uncorrected reference homography, the all-day average outperforms SIFT-FLANN, and the dynamic method outperforms the all-day average.

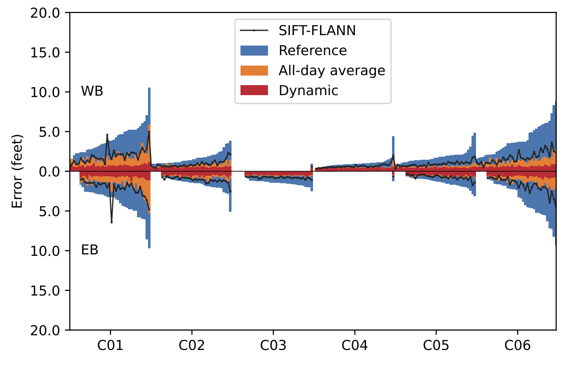


(a)

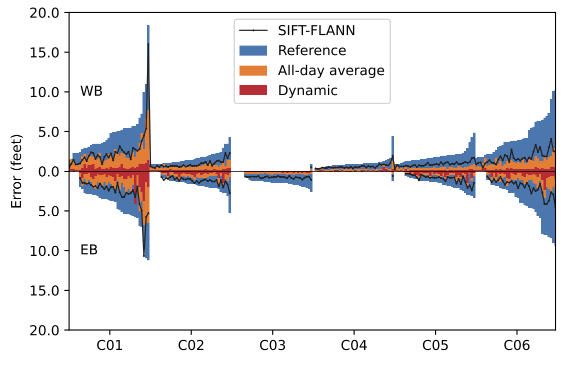


(b)

Figure Q.23: Error dynamics for two cameras over time with each homography re-estimation methods. Long field of view cameras e.g. Q.23a have more pronounced errors over down-looking cameras, e.g. Q.23b. Credit: Gergely Zachar.
.



(a) SubDrift



(b) FullDrift

Figure Q.24: Remaining SubDrift and FullDrift errors for each camera after (black) SIFT-FLANN feature-matching, (orange) one-day best fit homography re-estimation, and (red) dynamic homography re-estimation methods relative to original reference homography baseline (blue). Cameras are grouped by position on pole (see Figure 2 in main text.) and by side of roadway (westbound homographies on top, eastbound on bottom).) Credit: Gergely Zachar.

R Additional GPS Trajectory Plots for the I24-Video Dataset (Ch. 7)

Figure R.25 shows plots for additional GPS tracks through the video scene. Manually annotated object positions (circles) and GPS positions (lines) for are shown for the overall scene, divided by westbound (top left) and eastbound (bottom left) direction of travel. (right) x-position relative to the corrected GPS track (top) and absolute y-position (bottom) are plotted for the highlighted GPS track. It can be seen that a.) the corrected GPS trajectories align more closely with object detections (black dots) and b.) there is significant deviation between the corrected and uncorrected GPS trajectories. Note that especially the Y-coordinate error in the uncorrected trajectories varies in character across different GPS trajectories.

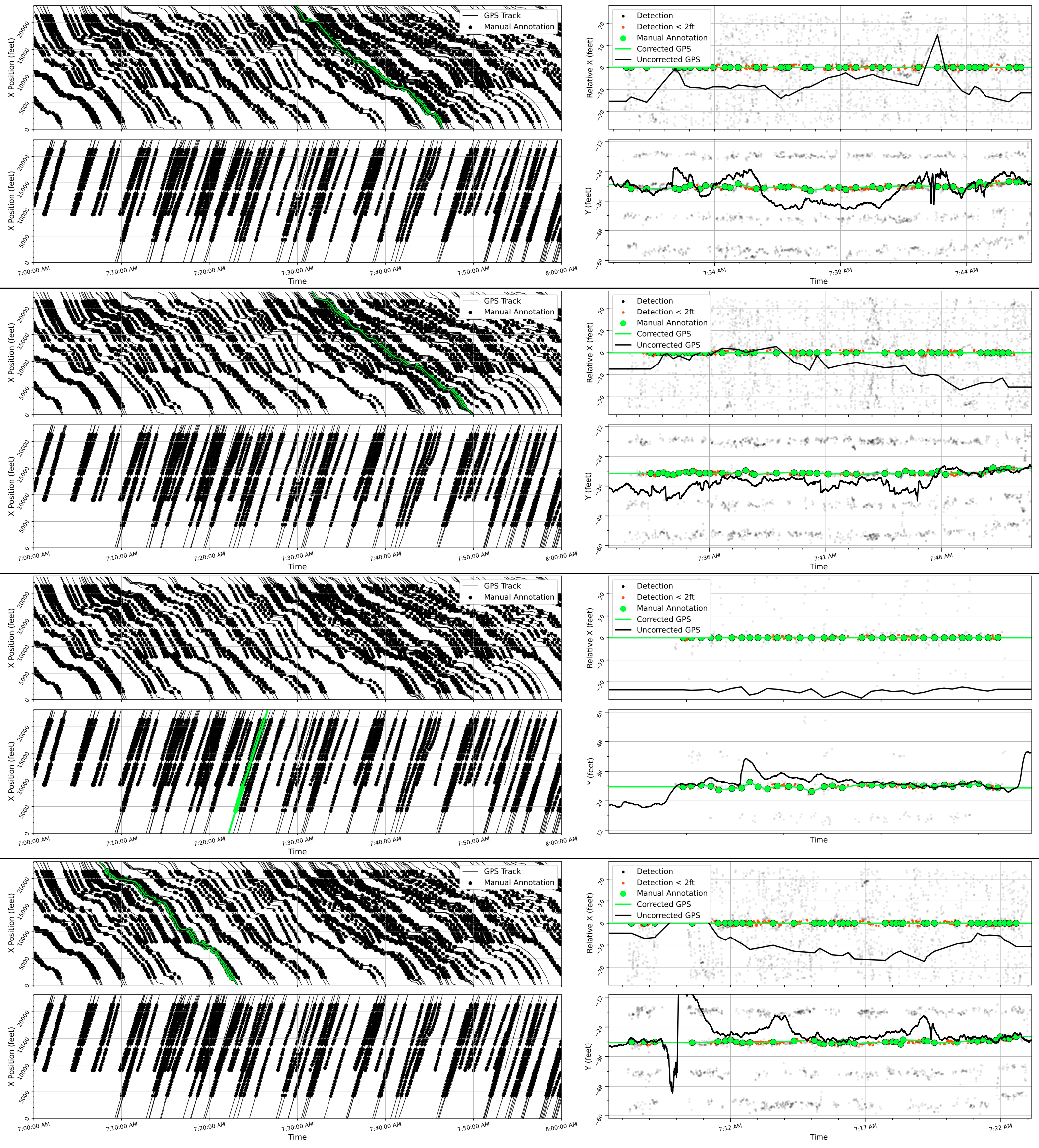


Figure R.25: Plots for 4 individual GPS trajectories. In each sub-figure, **(top left)** shows westbound X-position and **(bottom left)** shows eastbound X-position for the whole scene duration. **(top right)** shows nearby (in relative X-coordinates) and **(bottom right)** shows nearby (in Y-coordinates) detections, manual annotations, and uncorrected GPS trajectory, relative to the corrected GPS trajectory.

Appendix R is adapted from [93].

S Experimental Details for the I24-Video Dataset (Ch. 7)

S.1 Evaluation Protocol

Each object tracker is run using the detection set from Appendix 7.2.5. GPS trajectories and detections from each camera are obtained at slightly different times. To account for this, tracking evaluation is performed at fixed 0.1 second intervals, and each GPS trajectory and object tracklet position is linearly interpolated at each evaluation time. GPS trajectories extend somewhat outside of the temporal duration for which detections are available. Evaluation is performed only over the temporal range for which both GPS trajectories and detections exist. Moreover, each GPS trajectory is clipped in the X-range $[0, 23000]$ such that the trajectory is always visible within the field of view of the overall camera network. Evaluation is performed as in [216]. For all metrics other than HOTA metrics, a lax IOU of 0.1 is required for an object tracklet and GPS trajectory to be matched, and matching is performed with the Hungarian Algorithm for bipartite matching [307]. For each object tracklet, we use the median reported dimension (l,w, and h, in feet) over all reported tracklet dimension measurements.

S.2 Parameter Settings

Table 17 lists relevant parameter settings for each implemented algorithm. Kalman filter parameters were empirically fine-tuned using the I24-3D dataset [95].

Appendix S is adapted from [93].

Algorithm	Parameter	Value	Description
ALL	t_{max}	2	Maximum time (sec) between detections before track is terminated
	t_{min}	2	Minimum track length (sec)
	f_{eval}	10	Evaluation time step (Hz)
SORT [5]	σ_{high}	0.5	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	d_{max}	10	Maximum allowable distance for a match (ft)
	f_{track}	10	Tracking time step (Hz)
IOUT [166]	σ_{high}	0.5	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	ϕ_{min}	0.1	Minimum IOU for a match
	f_{track}	15	Tracking time step (Hz)
KIOU [322]	σ_{high}	0.5	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	ϕ_{min}	0.1	Minimum IOU for a match
	f_{track}	10	Tracking time step (Hz)
ByteTrack (L2) [226]	σ_{high}	0.01	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	d_{max}	10	Maximum allowable distance for a match (ft)
	τ_{high}	0.4	Required confidence to be included in first matching step
	f_{track}	10	Tracking time step (Hz)
ByteTrack (IOU) [226]	σ_{high}	0.01	Required object confidence to be included in detection set
	ϕ_{nms}	0.1	Non-maximal suppression IOU threshold
	ϕ_{min}	0.1	Minimum IOU for a match
	τ_{high}	0.4	Required confidence to be included in first matching step
	f_{track}	10	Tracking time step (Hz)
Oracle	ϕ_{min}	0.1	Minimum IOU for a match
	f_{track}	10	Tracking time step (Hz)

Table 17: Parameters for object tracking experiments on the I24-V dataset.

T Treatment of Personally Identifiable Information in the I24-Video Dataset (Ch. 7)

As stated in [34], the primary purpose of the I-24 system is not to produce video data, but rather is to produce vehicle trajectory data. The data management plan for the system states that in general, video data is not released to the public or stored for long periods of time, but occasionally video data may be released to enable work on training, testing, and validation of trajectory generation algorithms. Thus, the video released in this work represents an edge case for the system,

Nevertheless, the release of video data (or any data with individuals represented) carries with it the risk of releasing *personally identifiable information* (PII) on the included individuals. In this work we make every effort to prevent the leakage of PII to the public, and furthermore make it infeasible to automatically extract PII such that doing so becomes extremely onerous to potential bad actors. A three-tiered approach is used: 1.) We release, more or less, a random hour of data, such that the potential for capturing a discrete event of interest to a third party is near zero. 2.) We automatically redact all license plate information from all visible

Appendix T is adapted from [93].

vehicles. 3.) We manually redact any regions containing private property or visible people.

Even in un-redacted video data, license plates are in almost all cases impossible to read. Figure T.26 provides a typical camera field of view, with the license plate only about 15 pixels wide and subject to significant blurring from vehicle motion. (The vehicle used in this image is part of the GPS instrumented vehicle fleet and does not belong to an individual). Nevertheless, we run an off-the-shelf license plate redaction algorithm [347] on all frames and cover all detected license plates with a black rectangle.

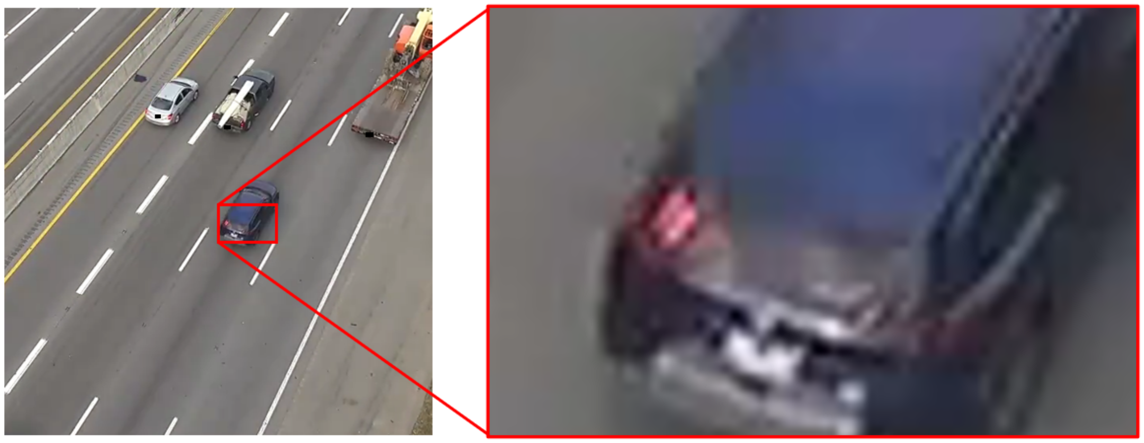


Figure T.26: Example license plate from this dataset before redaction. License plate information is unrecoverable; for reference, the orange numbers on the rear window of the vehicle are about 5 times as large as license plate text and are barely discernible. (The pictured vehicle is part of the GPS instrumented vehicle fleet and does not belong to an individual).

We then manually inspect each video and redact the following sets of information. These areas are blurred in the released videos. Polygons defining the redacted regions are also released such that data users can replace these pixel values as desired for computer vision applications. Figure T.27 shows an example of redacted regions.

- All visible people
- All private property
- Any stopped law enforcement vehicle (we leave ~ 1 sec of the stopped vehicle visible to allow for graceful handling of these vehicles by tracking algorithms.

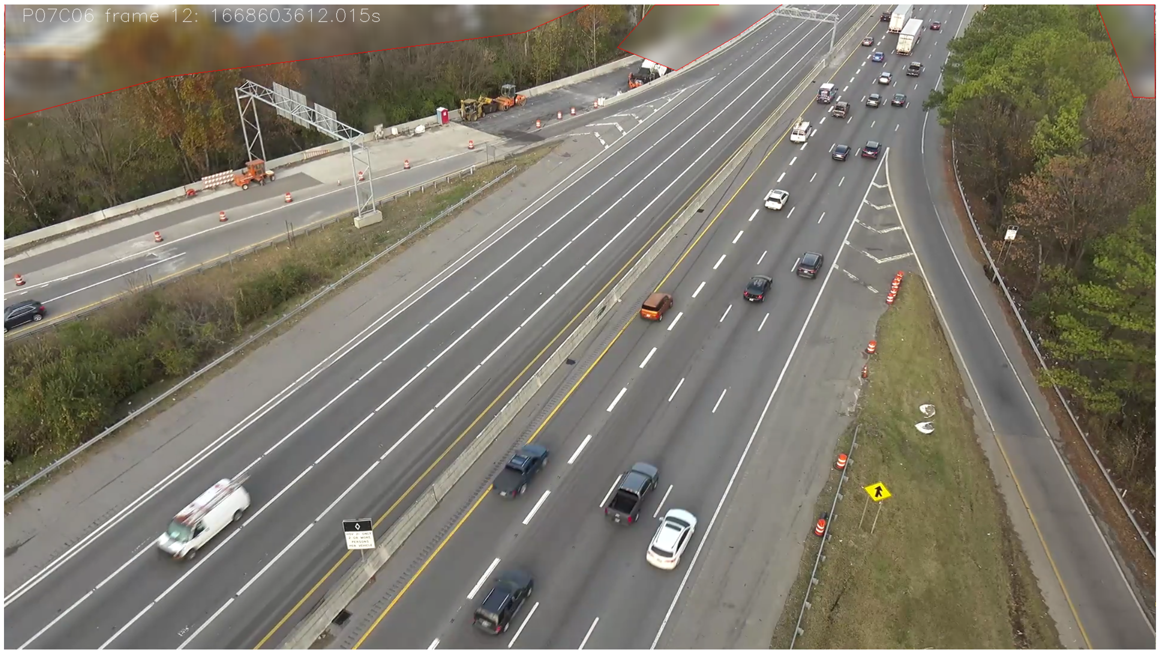


Figure T.27: Example redacted regions (red outline, blurred) for one camera field of view .

U Known Data Artifacts and Anomalies in the I24-Video Dataset (Ch. 7)

Here, we report a list of known anomalies and artifacts in the data as of submission time:

- Visible in some cameras on poles 3 and 4, there is a region that is blacked out in all frames and camera fields of view. This region corresponds to a private residence. Different from other redaction areas in this dataset, a virtual mask is applied to this region in all cameras in which it is visible, such that no visual information from this region is ever recorded. Other regions are redacted after recording.
- All cameras from Pole 25 are missing; this pole was struck by a vehicle a few days prior to the recording day and could not be restored in time.
- Homographies on the eastbound roadway side are not defined for cameras on poles 1-7. Construction work required that temporary solid lane markings be painted, for which i.) corner points were not uniquely distinguishable and ii.) no matching aerial imagery exists due to the short term nature of the construction work.
- Reference homographies for cameras on poles 1 and 2 are defined, but homographies are not re-estimated for these cameras. The lane markings were altered between the reference homography day and the recording day meaning that SIFT-FLANN matching and lane marking re-detection fail.
- Camera P22C04 has a black ring visible on the left portion of the frame due to a mechanical misalignment of the camera lens and body.
- Occasionally, GPS trajectories have missing recordings for time periods on the order of ~ 1 sec. The onboard sensor filtering attempts to compensate for this missing data, producing "sawtooth" artifacts in the trajectory. Figure U.28 shows an example. The majority of these artifacts were removed during data refinement, but some artifacts may still remain.

Appendix U is adapted from [93].

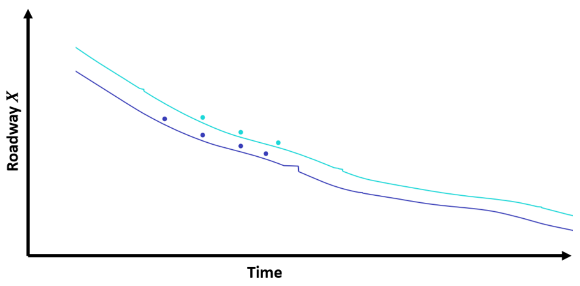


Figure U.28: “Sawtooth” artifact in uncorrected GPS trajectories (lines). Circles depict corresponding manually annotated vehicle positions.

References

[1] Siwei Lyu, Ming-Ching Chang, Dawei Du, Wenbo Li, Yi Wei, Marco Del Coco, Pierluigi Carcagnì, Arne Schumann, Bharti Munjal, Doo-Hyun Choi, et al. Ua-detrac 2018: Report of avss2018 & iwt4s challenge on advanced traffic monitoring. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[2] Leslie C Edie et al. *Discussion of traffic stream measurements and definitions*. Port of New York Authority New York, 1963.

[3] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 658–666, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.00075. URL https://ieeexplore.ieee.org/document/8953982/.

[4] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. IoU Loss for 2D/3D Object Detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94, Québec City, QC, Canada, September 2019. IEEE. ISBN 978-1-72813-131-3. doi: 10.1109/3DV.2019.00019. URL https://ieeexplore.ieee.org/document/8886046/.

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–308, 2009.

[7] Jorge A Laval and Carlos F Daganzo. Lane-changing in traffic streams. *Transportation Research Part B: Methodological*, 40(3):251–264, 2006.

[8] Joseph Treiterer and Jeffrey Myers. The hysteresis phenomenon in traffic flow. *Transportation and traffic theory*, 6:13–38, 1974.

[9] Yuki Sugiyama, Minoru Fukui, Macoto Kikuchi, Katsuya Hasebe, Akihiro Nakayama, Katsuhiro Nishinari, Shin-ichi Tadaki, and Satoshi Yukawa. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New journal of physics*, 10(3):033001, 2008.

[10] Raphael E Stern, Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, Hannah Pohlmann, Fangyu Wu, Benedetto Piccoli, et al. Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments. *Transportation Research Part C: Emerging Technologies*, 89:205–221, 2018.

[11] Dujuan Yang, Anique Kuijpers, Gamze Dane, and Tom van der Sande. Impacts of large-scale truck platooning on dutch highways. *Transportation research procedia*, 37:425–432, 2019.

[12] Nathan Lichtlé, Eugene Vinitsky, Matthew Nice, Benjamin Seibold, Dan Work, and Alexandre M Bayen. Deploying traffic smoothing cruise controllers learned from trajectory data. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2884–2890. IEEE, 2022.

[13] Matthew Bunting, Rahul Bhadani, and Jonathan Sprinkle. Libpanda: A high performance library for vehicle data collection. In *Proceedings of the Workshop on Data-Driven and Intelligent Cyber-Physical Systems*, DI-CPS'21, page 32–40, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384452. doi: 10.1145/3459609.3460529. URL https://doi.org/10.1145/3459609.3460529.

[14] Dongyao Jia and Dong Ngoduy. Enhanced cooperative car-following traffic model with the combination of v2v and v2i communication. *Transportation Research Part B: Methodological*, 90:172–191, 2016.

[15] Amir Ghiasi, Xiaopeng Li, and Jiaqi Ma. A mixed traffic speed harmonization model with connected autonomous vehicles. *Transportation Research Part C: Emerging Technologies*, 104:210–233, 2019.

[16] George Gunter, Derek Gloudemans, Raphael E Stern, Sean McQuade, Rahul Bhadani, Matt Bunting, Maria Laura Delle Monache, Roman Lysecky, Benjamin Seibold, Jonathan Sprinkle, et al. Are commercially implemented adaptive cruise control systems string stable? *IEEE Transactions on Intelligent Transportation Systems*, 22(11):6992–7003, 2020.

[17] Adolf Darlington May. *Traffic flow fundamentals*. Prentice Hall, 1990.

[18] Daniel S Turner. *75 Years of the Fundamental Diagram for Traffic Flow Theory: Greenshields Symposium: July 8-10, 2008, Woods Hole, Massachusetts*. Transportation Research Board, 2011.

[19] BD Greenshields, JR Bibbins, WS Channing, and HH Miller. A study of traffic capacity. In *Highway research board proceedings*, volume 1935. National Research Council (USA), Highway Research Board, 1935.

[20] Harold Greenberg. An analysis of traffic flow. *Operations research*, 7(1):79–85, 1959.

[21] Michael James Lighthill and Gerald Beresford Whitham. On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178):317–345, 1955.

[22] AATM Aw and Michel Rascle. Resurrection of" second order" models of traffic flow. *SIAM journal on applied mathematics*, 60(3):916–938, 2000.

[23] Roger P Roess, Elena S Prassas, and William R McShane. *Traffic engineering*. Pearson/Prentice Hall, 2004.

[24] Tom Choe, Alexander Skabardonis, and Pravin Varaiya. Freeway performance measurement system: operational analysis tool. *Transportation research record*, 1811(1):67–75, 2002.

[25] Martin Schönhof and Dirk Helbing. Empirical features of congested traffic states and their implications for traffic modeling. *Transportation Science*, 41(2):135–166, 2007.

[26] R Stewart, M Freeman, N Taylor, and D Fereday. Highways agency active traffic management: initial driver reactions to its implementation on the m42. In *PROCEEDINGS OF THE 13th ITS WORLD CONGRESS, LONDON, 8-12 OCTOBER 2006*, 2006.

[27] Hillel Bar-Gera. Evaluation of a cellular phone-based system for measurements of traffic speeds and travel times: A case study from israel. *Transportation Research Part C: Emerging Technologies*, 15 (6):380–391, 2007.

[28] Juan C Herrera, Daniel B Work, Ryan Herring, Xuegang Jeff Ban, Quinn Jacobson, and Alexandre M Bayen. Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment. *Transportation Research Part C: Emerging Technologies*, 18(4):568–583, 2010.

[29] Dirk Helbing. Empirical traffic data and their implications for traffic modeling. *Physical Review E*, 55 (1):R25, 1997.

[30] Dirk Helbing and Martin Treiber. Jams, waves, and clusters. *Science*, 282(5396):2001–2003, 1998.

[31] Boris S Kerner and Henry Lieu. The physics of traffic: Empirical freeway pattern features, engineering applications; and theory. *Physics Today*, 58(11):54–56, 2005.

[32] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[33] Zuduo Zheng, Soyoung Ahn, Danjue Chen, and Jorge Laval. Applications of wavelet transform for analysis of freeway traffic: Bottlenecks, transient traffic, and traffic oscillations. *Transportation Research Part B: Methodological*, 45(2):372–384, 2011.

[34] Derek Gloudemans, Yanbing Wang, Junyi Ji, Gergely Zachar, Will Barbour, and Daniel B Work. I-24 motion: An instrument for freeway traffic science. *arXiv preprint arXiv:2301.11198*, 2023.

[35] Robert E Chandler, Robert Herman, and Elliott W Montroll. Traffic dynamics: studies in car following. *Operations research*, 6(2):165–184, 1958.

[36] Arne Kesting and Martin Treiber. Calibrating car-following models by using trajectory data: Methodological study. *Transportation Research Record*, 2088(1):148–156, 2008.

[37] Willie D Jones. Keeping cars from crashing. *IEEE spectrum*, 38(9):40–45, 2001.

[38] Daniel Göhring, Miao Wang, Michael Schnürmacher, and Tinosch Ganjineh. Radar/lidar sensor fusion for car-following on highways. In *The 5th International Conference on Automation, Robotics and Applications*, pages 407–412. IEEE, 2011.

[39] Gemunu Senadeera Gurusinghe, Takashi Nakatsuji, Yoichi Azuta, Prakash Ranjitkar, and Yordphol Tanaboriboon. Multiple car-following data with real-time kinematic global positioning system. *Transportation Research Record*, 1802(1):166–180, 2002.

[40] Xiaoliang Ma and Ingmar Andréasson. Estimation of driver reaction time from car-following data: Application in evaluation of general motor–type model. *Transportation research record*, 1965(1): 130–141, 2006.

[41] Xiaopeng Li, Jianxun Cui, Shi An, and Mohsen Parsafard. Stop-and-go traffic analysis: Theoretical properties, environmental impacts and oscillation mitigation. *Transportation Research Part B: Methodological*, 70:319–339, 2014.

[42] Jorge A Laval and Ludovic Leclercq. A mechanism to describe the formation and propagation of stop-and-go waves in congested freeway traffic. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4519–4541, 2010.

[43] Antoine Tordeux, Sylvain Lassarre, and Michel Roussignol. An adaptive time gap car-following model. *Transportation research part B: methodological*, 44(8-9):1115–1131, 2010.

[44] Haris N Koutsopoulos and Haneen Farah. Latent class model for car following behavior. *Transportation research part B: methodological*, 46(5):563–578, 2012.

[45] Nachiket Deo and Mohan M Trivedi. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1179–1184. IEEE, 2018.

[46] Florent Altché and Arnaud de La Fortelle. An lstm network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, pages 353–359. IEEE, 2017.

[47] Hwasoo Yeo and Alexander Skabardonis. Understanding stop-and-go traffic in view of asymmetric traffic theory. In *Transportation and Traffic Theory 2009: Golden Jubilee*, pages 99–115. Springer, 2009.

[48] Leslie C Edie and Robert S Foote. Traffic flow in tunnels. In *Highway Research Board Proceedings*, volume 37, 1958.

[49] Vassili Alexiadis, James Colyar, John Halkias, Rob Hranac, and Gene McHale. The next generation simulation program. *Institute of Transportation Engineers. ITE Journal*, 74(8):22, 2004.

[50] Benjamin Coifman and Lizhe Li. A critical evaluation of the next generation simulation (ngsim) vehicle trajectory dataset. *Transportation Research Part B: Methodological*, 105:362–377, 2017.

[51] T Kim and HM Zhang. A stochastic wave propagation model. *Transportation Research Part B: Methodological*, 42(7-8):619–634, 2008.

[52] Martin Treiber and Arne Kesting. Evidence of convective instability in congested traffic flow: A systematic empirical and theoretical investigation. *Procedia-Social and Behavioral Sciences*, 17:683–701, 2011.

[53] Jie Sun, Zuduo Zheng, and Jian Sun. The relationship between car following string instability and traffic oscillations in finite-sized platoons and its use in easing congestion via connected and automated vehicles with idm based controller. *Transportation Research Part B: Methodological*, 142:58–83, 2020.

[54] Li Li, Rui Jiang, Zhengbing He, Xiqun Michael Chen, and Xuesong Zhou. Trajectory data-based traffic flow studies: A revisit. *Transportation Research Part C: Emerging Technologies*, 114:225–240, 2020.

[55] Toru Seo, Alexandre M Bayen, Takahiko Kusakabe, and Yasuo Asakura. Traffic state estimation on highway: A comprehensive survey. *Annual reviews in control*, 43:128–151, 2017.

[56] S. I. Khan and P. Maini. Modeling heterogeneous traffic flow. *Transportation Research Record: Journal of the Transportation Research Board*, 1678:234–241, 1999.

[57] V. T. Arasan and R. Z. Koshy. Methodology for modeling highly heterogeneous traffic flow. *Journal of Transportation Engineering*, 131:544, 2005.

[58] Lasmini Ambarwati, Adam J Pel, Robert Verhaeghe, and Bart van Arem. Empirical analysis of heterogeneous traffic flow and calibration of porous flow model. *Transportation research part C: emerging technologies*, 48:418–436, 2014.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[61] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[62] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[63] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578, 2019.

[64] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[65] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[66] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE, 2018.

[67] Tobias Moers, Lennart Vater, Robert Krajewski, Julian Bock, Adrian Zlocki, and Lutz Eckstein. The exid dataset: A real-world trajectory dataset of highly interactive highway scenarios in germany. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 958–964, 2022. doi: 10.1109/IV51971.2022.9827305.

[68] Paul Spannaus, Peter Zechel, and Kilian Lenz. Automatum data: Drone-based highway dataset for the development and validation of automated driving software for research and commercial applications. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 1372–1377. IEEE, 2021.

[69] Xiaowei Shi, Dongfang Zhao, Handong Yao, Xiaopeng Li, David K Hale, and Amir Ghiasi. Video-based trajectory extraction with deep learning for high-granularity highway simulation (high-sim). *Communications in transportation research*, 1:100014, 2021.

[70] Rachel James. Third generation simulation: A closer look at the impact of automated driving systems on traffic, 2023. URL https://highways.dot.gov/research/projects/third-generation-simulation-closer-look-impact-automated-driving-systems-traffic.

[71] Emmanouil Barmpounakis and Nikolas Geroliminis. On the new era of urban traffic monitoring with massive drone data: The pneuma large-scale field experiment. *Transportation research part C: emerging technologies*, 111:50–71, 2020.

[72] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1929–1934. IEEE, 2020.

[73] Robert Krajewski, Tobias Moers, Julian Bock, Lennart Vater, and Lutz Eckstein. The round dataset: A drone dataset of road user trajectories at roundabouts in germany. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020. doi: 10.1109/ITSC45102.2020.9294728.

[74] Antonia Breuer, Jan-Aike Termöhlen, Silviu Homoceanu, and Tim Fingscheidt. opendd: A large-scale roundabout drone dataset. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[75] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

[76] Ou Zheng, Mohamed Abdel-Aty, Lishengsa Yue, Amr Abdelraouf, Zijin Wang, and Nada Mahmoud. Citysim: A drone-based vehicle trajectory dataset for safety oriented research and digital twins. *arXiv preprint arXiv:2208.11036*, 2022.

[77] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/He_Mask_R-CNN_ICCV_2017_paper.html.

[78] N. Strege, P. Nelson, A. Friebe, K. Seppanen, and M.L. Schier. *Annual Report*. University of Minnesota, Center for Transportation Studies, 200 Transportation and Safety Building, 511 Washington Ave. S.E., Minneapolis, MN 55455, 2004.

[79] Antje von Schmidt, María López Díaz, and Alain Schengen. Creating a baseline scenario for simulating travel demand: A case study for preparing the region test bed lower saxony, germany. In *International Conference on Advances in System Simulation (SIMUL)*, pages 51–57. ThinkMind, 2021.

[80] Toru Seo, Yusuke Tago, Norihito Shinkai, Masakazu Nakanishi, Jun Tanabe, Daisuke Ushirogochi, Shota Kanamori, Atsushi Abe, Takashi Kodama, Satoshi Yoshimura, et al. Evaluation of large-scale complete vehicle trajectories dataset on two kilometers highway segment for one hour duration: Zen traffic data. In *2020 International Symposium on Transportation Data and Modelling*, 2020.

[81] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[82] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv:2003.09003[cs]*, March 2020. URL http://arxiv.org/abs/1906.04567. arXiv: 2003.09003.

[83] Cathy Wu, Alexandre M Bayen, and Ankur Mehta. Stabilizing traffic with autonomous vehicles. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6012–6018. IEEE, 2018.

[84] Derek Gloudemans, William Barbour, Nikki Gloudemans, Matthew Neuendorf, Brad Freeze, Said ElSaid, and Daniel B Work. Interstate-24 motion: Closing the loop on smart mobility. In *2020 IEEE Workshop on Design Automation for CPS and IoT (DESTION)*, pages 49–55. IEEE, 2020.

[85] W. Barbour, D. Gloudemans, M. Cebelak, P. Freeze, and D. Work. Interstate 24 motion open road testbed. In *Proceedings of the ITS America Annual Meeting, to appear*, location, 12 2021.

[86] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 941–951, 2019.

[87] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

[88] Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li, and Cewu Lu. Tubetk: Adopting tubes to track multi-object in a one-step training model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6308–6318, 2020.

[89] Shyamgopal Karthik, Ameya Prabhu, and Vineet Gandhi. Simple unsupervised multi-object tracking. *arXiv preprint arXiv:2006.02609*, 2020.

[90] Wei Li, Yuanjun Xiong, Shuo Yang, Siqi Deng, and Wei Xia. Smot: Single-shot multi object tracking. *arXiv preprint arXiv:2010.16031*, 2020.

[91] Derek Gloudemans and Daniel B Work. Vehicle tracking with crop-based detection. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 312–319. IEEE, 2021.

[92] Derek Gloudemans and Daniel B Work. Fast vehicle turning-movement counting using localization-based tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4155–4164, 2021.

[93] Derek Gloudemans, Gergely Zachar, Yanbing Wang, Junyi Ji, William Barbour, Matthew Nice, Matthew Bunting, Jonathan Sprinkle, Maria Laura Delle Monache, Benedetto Piccoli, Alexandre Bayen, Benjamin Seibold, and Daniel B. Work. So you think you can track? *arXiv preprint arXiv:2309.07268*, 2023.

[94] Derek Gloudemans, Xinxuan Lu, Shepard Xia, and Daniel B. Work. Polygon intersection-over-union loss for viewpoint-agnostic monocular 3d vehicle detection. *arXiv preprint arXiv:2309.07104*, 2023.

[95] Derek Gloudemans, Gracie Gumm, Yanbing Wang, William Barbour, and Daniel B. Work. The interstate-24 3d dataset: a new benchmark for 3d multi-camera vehicle tracking. *arXiv preprint arXiv:2308.14833*, 2023.

[96] Azadeh Emami, Majid Sarvi, and Saeed Asadi Bagloee. A review of the critical elements and development of real-world connected vehicle testbeds around the world. *Transportation Letters*, pages 1–26, 2020.

[97] American Center for Mobility. Mobility research. Online, accessed April 2021, `https://www.acmwillowrun.org/`, 2021.

[98] UMTRI Briefs. Mcity grand opening. *Research Review*, 46(3), 2015.

[99] Akansel Cosgun, Lichao Ma, Jimmy Chiu, Jiawei Huang, Mahmut Demir, Alexandre Miranda Anon, Thang Lian, Hasan Tafish, and Samir Al-Stouhi. Towards full automated drive in urban environments: A demonstration in gomentum station, california. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1811–1818. IEEE, 2017.

[100] Fred Heery Sr et al. The florida connected and automated vehicle initiative: a focus on deployment. *Institute of Transportation Engineers. ITE Journal*, 87(10):33–41, 2017.

[101] Gordon Parikh and John Hourdos. Implementation of high accuracy radar detectors for traffic safety countermeasure evaluation, 2014.

[102] Ray C. Anderson Foundation. Welcome to The Ray. Online, accessed April 2021, `https://theray.org/technology/`, 2021.

[103] J. Farrell and M.J. Barth et al. Precision mapping of the california connected vehicle testbed corridor. Technical Report CA16-2777, California. Dept. of Transportation, 2015.

[104] University of Michigan Engineering. About Ann Arbor Connected Vehicle Test Environment (AACVTE). Online, accessed April 2021, `https://aacvte.engin.umich.edu`, 2021.

[105] Annkathrin Krämmer, Christoph Schöller, Dhiraj Gulati, and Alois Knoll. Providentia-a large scale sensing system for the assistance of autonomous vehicles. In *Robotics: Science and Systems (RSS), Workshop on Scene and Situation Understanding for Autonomous Driving*, 2019.

[106] FHWA. West central alabama action, 2022. https://ops.fhwa.dot.gov/fastact/atcmtd/2017/applications/univalabama/project.htm.

[107] Yiqun Dong, Yuanxin Zhong, Wenbo Yu, Minghan Zhu, Pingping Lu, Yeyang Fang, Jiajun Hong, and Huei Peng. Mcity data collection for automated vehicles study. *arXiv preprint arXiv:1912.06258*, 2019.

[108] Ray C. Anderson Foundation. The ray, 2014. https://theray.org/.

[109] CalTrans. California connected vehicle testbed, 2019. https://caconnectedvehicletestbed.org/.

[110] American Center for Mobility. Acm willow run, 2017. https://www.acmwillowrun.org/.

[111] Mathieu Joerger, Cynthia Jones, and Valerie Shuman. Testing connected and automated vehicles (cavs): Accelerating innovation, integration, deployment and sharing results. In *Road Vehicle Automation 5*, pages 197–206. Springer, 2019.

[112] Mashrur Chowdhury, Mizanur Rahman, Anjan Rayamajhi, Sakib Mahmud Khan, Mhafuzul Islam, Zadid Khan, and James Martin. Lessons learned from the real-world deployment of a connected vehicle testbed. *Transportation Research Record*, 2672(22):10–23, 2018.

[113] Markéta Dubská, Adam Herout, and Jakub Sochor. Automatic camera calibration for traffic understanding. In *BMVC*, volume 4, page 8, 2014.

[114] Markéta Dubská, Adam Herout, Roman Juránek, and Jakub Sochor. Fully automatic roadside camera calibration for traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 16:1162–1171, 2014.

[115] Jakub Sochor, Jakub Špaňhel, and Adam Herout. Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance. *IEEE transactions on intelligent transportation systems*, 20(1):97–108, 2018.

[116] Xinhe Ren, David Wang, Michael Laskey, and Ken Goldberg. Learning traffic behaviors by extracting vehicle trajectories from online video streams. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 1276–1283. IEEE, 2018.

[117] 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.

[118] Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang. Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 108–115, 2018.

[119] Yucheng Chen, Longlong Jing, Elahe Vahdani, Ling Zhang, Mingyi He, and Yingli Tian. Multi-camera vehicle tracking and re-identification on ai city challenge 2019. In *CVPR Workshops*, volume 2, pages 324–332, 2019.

[120] Dongfang Zhao and Xiaopeng Li. Real-world trajectory extraction from aerial videos-a comprehensive and effective solution. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2854–2859. IEEE, 2019.

[121] Tianya Zhang and Peter J Jin. A longitudinal scanline based vehicle trajectory reconstruction method for high-angle traffic video. *Transportation research part C: emerging technologies*, 103:104–128, 2019.

[122] Yegor Malinovskiy, Yao-Jan Wu, and Yinhai Wang. Video-based vehicle detection and tracking using spatiotemporal maps. *Transportation research record*, 2121(1):81–89, 2009.

[123] Jonathan M Hankey, Miguel A Perez, and Julie A McClafferty. Description of the shrp 2 naturalistic database and the crash, near-crash, and baseline data sets. Technical report, Virginia Tech Transportation Institute, 2016.

[124] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[125] Siyuan Liu, Ce Liu, Qiong Luo, Lionel M Ni, and Ramayya Krishnan. Calibrating large scale vehicle trajectory data. In *2012 IEEE 13th International Conference on Mobile Data Management*, pages 222–231. IEEE, 2012.

[126] Wenhuan Shi, Shuhan Shen, and Yuncai Liu. Automatic generation of road network map from massive gps, vehicle trajectories. In *2009 12th international IEEE conference on intelligent transportation systems*, pages 1–6. IEEE, 2009.

[127] Gunwoo Lee, Soyoung You, Stephen G Ritchie, Jean-Daniel Saphores, Mana Sangkapichai, and R Jayakrishnan. Environmental impacts of a major freight corridor: a study of i-710 in california. *Transportation Research Record*, 2123(1):119–128, 2009.

[128] Jian Sun, Han Liu, and Zian Ma. Modelling and simulation of highly mixed traffic flow on two-lane two-way urban streets. *Simulation Modelling Practice and Theory*, 95:16–35, 2019.

[129] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC)*, pages 2575–2582. IEEE, 2018.

[130] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *CoRR*, abs/1804.02767, 2018. URL http://arxiv.org/abs/1804.02767. arXiv: 1804.02767.

[131] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection, April 2020. URL http://arxiv.org/abs/2004.10934. arXiv:2004.10934 [cs, eess].

[132] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[133] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[134] Kaiwen Duan, Lingxi Xie, Honggang Qi, Song Bai, Qingming Huang, and Qi Tian. Corner proposal network for anchor-free, two-stage object detection. *arXiv preprint arXiv:2007.13816*, 2020.

[135] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[136] Li Wang, Yao Lu, Hong Wang, Yingbin Zheng, Hao Ye, and Xiangyang Xue. Evolving boxes for fast vehicle detection. In *2017 IEEE international conference on multimedia and Expo (ICME)*, pages 1135–1140. IEEE, 2017.

[137] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[138] Hughes Perreault, Guillaume-Alexandre Bilodeau, Nicolas Saunier, and Maguelonne Héritier. Spotnet: Self-attention multi-task network for object detection. In *2020 17th Conference on Computer and Robot Vision (CRV)*, pages 230–237. IEEE, 2020.

[139] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[140] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[141] Zhihang Fu, Yaowu Chen, Hongwei Yong, Rongxin Jiang, Lei Zhang, and Xian-Sheng Hua. Foreground gating and background refining network for surveillance object detection. *IEEE Transactions on Image Processing*, 28(12):6077–6090, 2019.

[142] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3D Object Detection for Autonomous Driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016. URL https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Chen_Monocular_3D_Object_CVPR_2016_paper.html.

[143] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018.

[144] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5390–5399, 2019.

[145] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5632–5640, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.597. URL http://ieeexplore.ieee.org/document/8100080/.

[146] Andretti Naiden, Vlad Paunescu, Gyeongmo Kim, ByeongMoon Jeon, and Marius Leordeanu. Shift R-CNN: Deep Monocular 3D Object Detection With Closed-Form Geometric Constraints. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 61–65, September 2019. doi: 10.1109/ICIP.2019.8803397. ISSN: 2381-8549.

[147] Jiaojiao Fang, Lingtao Zhou, and Guizhong Liu. 3D Bounding Box Estimation for Autonomous Vehicles by Cascaded Geometric Constraints and Depurated 2D Detections Using 3D Results, September 2019. URL http://arxiv.org/abs/1909.01867. arXiv:1909.01867 [cs].

[148] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teuliere, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2040–2049, 2017.

[149] Abhijit Kundu, Yin Li, and James M. Rehg. 3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3559–3568, Salt Lake City, UT, USA, June 2018. IEEE. ISBN 978-1-5386-6420-9. doi: 10.1109/CVPR.2018.00375. URL https://ieeexplore.ieee.org/document/8578473/.

[150] Tong He and Stefano Soatto. Mono3D++: Monocular 3D Vehicle Detection with Two-Scale 3D Hypotheses and Task Priors. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 8409–8416, July 2019. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.33018409. URL https://ojs.aaai.org/index.php/AAAI/article/view/4856. Number: 01.

[151] Xuepeng Shi, Qi Ye, Xiaozhi Chen, Chuangrong Chen, Zhixiang Chen, and Tae-Kyun Kim. Geometry-based Distance Decomposition for Monocular 3D Object Detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15152–15161, Montreal, QC, Canada, October 2021. IEEE. ISBN 978-1-66542-812-5. doi: 10.1109/ICCV48922.2021.01489. URL https://ieeexplore.ieee.org/document/9711219/.

[152] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. RTM3D: Real-Time Monocular 3D Detection from Object Keypoints for Autonomous Driving. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 644–660, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58580-8. doi: 10.1007/978-3-030-58580-8_38.

[153] Xianpeng Liu, Nan Xue, and Tianfu Wu. Learning Auxiliary Monocular Contexts Helps Monocular 3D Object Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 1810–1818, June 2022. doi: 10.1609/aaai.v36i2.20074. URL https://ojs.aaai.org/index.php/AAAI/article/view/20074. ISSN: 2374-3468, 2159-5399 Issue: 2 Journal Abbreviation: AAAI.

[154] Yunpeng Zhang, Jiwen Lu, and Jie Zhou. Objects are Different: Flexible Monocular 3D Object Detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3288–3297, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.00330. URL https://ieeexplore.ieee.org/document/9578273/.

[155] Jakub Sochor, Adam Herout, and Jiri Havel. Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3006–3015, 2016.

[156] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[157] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *Proceedings of the AAAI Conference on*

Artificial Intelligence, 34(07):12993–13000, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i07.6999. URL https://ojs.aaai.org/index.php/AAAI/article/view/6999. Number: 07.

[158] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation. *IEEE Transactions on Cybernetics*, 52(8):8574–8586, August 2022. ISSN 2168-2275. doi: 10.1109/TCYB.2021.3095305. Conference Name: IEEE Transactions on Cybernetics.

[159] Yu Zheng, Danyang Zhang, Sinan Xie, Jiwen Lu, and Jie Zhou. Rotation-robust intersection over union for 3d object detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX*, pages 464–480. Springer, 2020.

[160] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, Providence, RI, June 2012. IEEE. ISBN 978-1-4673-1228-8 978-1-4673-1226-4 978-1-4673-1227-1. doi: 10.1109/CVPR.2012.6248074. URL http://ieeexplore.ieee.org/document/6248074/.

[161] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[162] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381: 61–88, 2020.

[163] Litong Fan, Zhongli Wang, Baigen Cail, Chuanqi Tao, Zhiyi Zhang, Yinling Wang, Shanwen Li, Fengtian Huang, Shuangfu Fu, and Feng Zhang. A survey on multiple object tracking algorithm. In *2016 IEEE International Conference on Information and Automation (ICIA)*, pages 1855–1862. IEEE, 2016.

[164] Jiahui Chen, Hao Sheng, Yang Zhang, and Zhang Xiong. Enhancing detection model for multiple hypothesis tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 18–27, 2017.

[165] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 4696–4704, 2015.

[166] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[167] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv e-prints*, pages arXiv–2004, 2020.

[168] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. Transmot: Spatial-temporal graph transformer for multiple object tracking. *arXiv preprint arXiv:2104.00194*, 2021.

[169] Ioannis Papakis, Abhijit Sarkar, and Anuj Karpatne. Gcnnmatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization. *arXiv preprint arXiv:2010.00067*, 2020.

[170] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *European Conference on Computer Vision*, pages 107–122. Springer, 2020.

[171] Zheng Zhang, Dazhi Cheng, Xizhou Zhu, Stephen Lin, and Jifeng Dai. Integrated object detection and tracking with tracklet-conditioned detection. *arXiv preprint arXiv:1811.11167*, 2018.

[172] M Jaward, L Mihaylova, N Canagarajah, and D Bull. Multiple object tracking using particle filters. In *2006 IEEE Aerospace Conference*, pages 8–pp. IEEE, 2006.

[173] Cheng Chang, Rashid Ansari, and Ashfaq Khokhar. Multiple object tracking with kernel particle filter. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 566–573. IEEE, 2005.

[174] Jerome Berclaz, Francois Fleuret, and Pascal Fua. Multiple object tracking using flow linear programming. In *2009 Twelfth IEEE international workshop on performance evaluation of tracking and surveillance*, pages 1–8. IEEE, 2009.

[175] Peng Dai, Renliang Weng, Wongun Choi, Changshui Zhang, Zhangping He, and Wei Ding. Learning a proposal classifier for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2443–2452, 2021.

[176] Michael Engquist. A successive shortest path algorithm for the assignment problem. *INFOR: Information Systems and Operational Research*, 20(4):370–384, 1982.

[177] William Brendel, Mohamed Amer, and Sinisa Todorovic. Multiobject tracking as maximum weight independent set. In *CVPR 2011*, pages 1273–1280. IEEE, 2011.

[178] Bo Yang, Chang Huang, and Ram Nevatia. Learning affinities and dependencies for multi-target tracking using a crf model. In *CVPR 2011*, pages 1233–1240. IEEE, 2011.

[179] Peng Chu and Haibin Ling. Famnet: Joint learning of feature, affinity and multi-dimensional assignment for online multiple object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6172–6181, 2019.

[180] Bastian Leibe, Konrad Schindler, Nico Cornelis, and Luc Van Gool. Coupled object detection and tracking from static cameras and moving vehicles. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1683–1698, 2008.

[181] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):58–72, 2013.

[182] Paula Craciun, Mathias Ortner, and Josiane Zerubia. Joint detection and tracking of moving objects using spatio-temporal marked point processes. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 177–184. IEEE, 2015.

[183] Anton Andriyenko and Konrad Schindler. Multi-target tracking by continuous energy minimization. In *CVPR 2011*, pages 1265–1272. IEEE, 2011.

[184] Anton Milan, Konrad Schindler, and Stefan Roth. Detection-and trajectory-level exclusion in multiple object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3682–3689, 2013.

[185] Anton Milan, Laura Leal-Taixé, Konrad Schindler, and Ian Reid. Joint tracking and segmentation of multiple targets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5397–5406, 2015.

[186] Chanho Kim, Fuxin Li, and James M Rehg. Multi-object tracking with neural gating using bilinear lstm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 200–215, 2018.

[187] Xinchao Wang, Bin Fan, Shiyu Chang, Zhangyang Wang, Xianming Liu, Dacheng Tao, and Thomas S Huang. Greedy batch-based minimum-cost flows for tracking multiple objects. *IEEE Transactions on Image Processing*, 26(10):4765–4776, 2017.

[188] Congchao Wang, Yizhi Wang, Yinxue Wang, Chiung-Ting Wu, and Guoqiang Yu. mussp: Efficient min-cost flow algorithm for multi-object tracking. *Advances in Neural Information Processing Systems*, 32, 2019.

[189] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6247–6257, 2020.

[190] Chee-Yee Chong. Graph approaches for data association. In *2012 15th International Conference on Information Fusion*, pages 1578–1585. IEEE, 2012.

[191] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE international conference on computer vision*, pages 4705–4713, 2015.

[192] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12352–12361, 2021.

[193] Zhichao Lu, Vivek Rathod, Ronny Votel, and Jonathan Huang. Retinatrack: Online single stage joint detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14668–14678, 2020.

[194] Bharti Munjal, Abdul Rafey Aftab, Sikandar Amin, Meltem D Brandlmaier, Federico Tombari, and Fabio Galasso. Joint detection and tracking in videos with identification features. *Image and Vision Computing*, 100:103932, 2020.

[195] ShiJie Sun, Naveed Akhtar, HuanSheng Song, Ajmal Mian, and Mubarak Shah. Deep affinity network for multiple object tracking. *IEEE transactions on pattern analysis and machine intelligence*, 43(1): 104–119, 2019.

[196] Qiang Wang, Yun Zheng, Pan Pan, and Yinghui Xu. Multiple object tracking with correlation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3876–3886, 2021.

[197] Cong Ma, Fan Yang, Yuan Li, Huizhu Jia, Xiaodong Xie, and Wen Gao. Deep human-interaction and association by graph-based learning for multiple object tracking in the wild. *International Journal of Computer Vision*, 129(6):1993–2010, 2021.

[198] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. Online video object detection using association lstm. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2344–2352, 2017.

[199] Yiming Liang and Yue Zhou. Lstm multiple object tracker combining multiple cues. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2351–2355. IEEE, 2018.

[200] Hilke Kieritz, Wolfgang Hubner, and Michael Arens. Joint detection and online multi-object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1459–1467, 2018.

[201] Anton Milan, S Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[202] Yongxin Wang, Xinshuo Weng, and Kris Kitani. Joint detection and multi-object tracking with graph neural networks. *arXiv preprint arXiv:2006.13164*, 2020.

[203] Juan Diego Gonzales Zuniga, Ujjwal Ujjwal, and Francois Bremond. Detracker: A joint detection and tracking framework. In *VISSAP 2022-International Conference on Computer Vision Theory and Applications*, 2022.

[204] Shibo Zhang and Xiaojie Wang. Human detection and object tracking based on histograms of oriented gradients. In *2013 ninth international conference on natural computation (ICNC)*, pages 1349–1353. IEEE, 2013.

[205] Chao Liang, Zhipeng Zhang, Yi Lu, Xue Zhou, Bing Li, Xiyong Ye, and Jianxiao Zou. Rethinking the competition between detection and reid in multi-object tracking. *arXiv preprint arXiv:2010.12138*, 2020.

[206] Yifu Zhan, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. A simple baseline for multi-object tracking. *arXiv preprint arXiv:2004.01888*, 2020.

[207] Jinlong Peng, Changan Wang, Fangbin Wan, Yang Wu, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking. In *European Conference on Computer Vision*, pages 145–161. Springer, 2020.

[208] Chenge Li, Gregory Dobler, Xin Feng, and Yao Wang. Tracknet: Simultaneous object detection and tracking and its application in traffic video analysis. *arXiv preprint arXiv:1902.01466*, 2019.

[209] Peng Chu, Heng Fan, Chiu C Tan, and Haibin Ling. Online multi-object tracking with instance-aware tracker and dynamic model refreshment. In *2019 IEEE winter conference on applications of computer vision (WACV)*, pages 161–170. IEEE, 2019.

[210] Weitao Feng, Zhihao Hu, Wei Wu, Junjie Yan, and Wanli Ouyang. Multi-object tracking with multiple cues and switcher-aware classification. *arXiv preprint arXiv:1901.06129*, 2019.

[211] Xu Yan, Xuqing Wu, Ioannis A Kakadiaris, and Shishir K Shah. To track or to detect? an ensemble framework for optimal selection. In *European Conference on Computer Vision*, pages 594–607. Springer, 2012.

[212] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4836–4845, 2017.

[213] ShiJie Sun, Naveed Akhtar, XiangYu Song, HuanSheng Song, Ajmal Mian, and Mubarak Shah. Simultaneous detection and tracking with motion modelling for multiple object tracking. *arXiv preprint arXiv:2008.08826*, 2020.

[214] Erik Bochinski, Tobias Senst, and Thomas Sikora. Extending iou based multi-object tracking by visual information. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[215] Longyin Wen, Dawei Du, Zhaowei Cai, Zhen Lei, Ming-Ching Chang, Honggang Qi, Jongwoo Lim, Ming-Hsuan Yang, and Siwei Lyu. Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, 193:102907, 2020.

[216] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[217] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[218] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129(2):548–578, 2021.

[219] En Yu, Zhuoling Li, Shoudong Han, and Hongwei Wang. Relationtrack: Relation-aware multiple object tracking with decoupled representation. *arXiv preprint arXiv:2105.04322*, 2021.

[220] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. Transcenter: Transformers with dense queries for multiple-object tracking. *arXiv preprint arXiv:2103.15145*, 2021.

[221] Wei Li, Yuanjun Xiong, Shuo Yang, Mingze Xu, Yongxin Wang, and Wei Xia. Semi-tcl: Semi-supervised track contrastive representation learning. *arXiv preprint arXiv:2107.02396*, 2021.

[222] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046, 2017.

[223] Qi Chu, Wanli Ouyang, Bin Liu, Feng Zhu, and Nenghai Yu. Dasot: A unified framework integrating data association and single object tracking for online multi-object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10672–10679, 2020.

[224] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702*, 2021.

[225] Fangao Zeng, Bin Dong, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. *arXiv preprint arXiv:2105.03247*, 2021.

[226] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 1–21. Springer, 2022.

[227] Andrei Barbu, Aaron Michaux, Siddharth Narayanaswamy, and Jeffrey Mark Siskind. Simultaneous object detection, tracking, and event recognition. *arXiv preprint arXiv:1204.2741*, 2012.

[228] Sam Hare, Stuart Golodetz, Amir Saffari, Vibhav Vineet, Ming-Ming Cheng, Stephen L Hicks, and Philip HS Torr. Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2096–2109, 2015.

[229] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[230] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. A twofold siamese network for real-time object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4834–4843, 2018.

[231] Jingjing Xiao, Rustam Stolkin, and Ales Leonardis. Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4978–4987, 2015.

[232] Jae Kyu Suhr. Kanade-lucas-tomasi (klt) feature tracker. *Computer Vision (EEE6503)*, pages 9–18, 2009.

[233] Ryota Yoshihashi, Rei Kawakami, Shaodi You, Tu Tuan Trinh, Makoto Iida, and Takeshi Naemura. Finding a needle in a haystack: Tiny flying object detection in 4k videos using a joint detection-and-tracking approach. *arXiv preprint arXiv:2105.08253*, 2021.

[234] Manchen Wang, Joseph Tighe, and Davide Modolo. Combining detection and tracking for human pose estimation in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11088–11096, 2020.

[235] Bin Wang, Sheng Tang, Jun-Bin Xiao, Quan-Feng Yan, and Yong-Dong Zhang. Detection and tracking based tubelet generation for video object detection. *Journal of Visual Communication and Image Representation*, 58:102–111, 2019.

[236] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple-object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020.

[237] Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez, Roberto López-Sastre, Saturnino Maldonado-Bascón, and Daniel Onoro-Rubio. Extremely overlapping vehicle counting. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 423–431. Springer, 2015.

[238] Erhan Bas, A Murat Tekalp, and F Sibel Salman. Automatic vehicle counting from video for traffic flow analysis. In *2007 IEEE intelligent vehicles symposium*, pages 392–397. Ieee, 2007.

[239] Yingjie Xia, Xingmin Shi, Guanghua Song, Qiaolei Geng, and Yuncai Liu. Towards improving quality of video-based vehicle counting method for traffic flow estimation. *Signal Processing*, 120:672–681, 2016.

[240] Z. Al-Ariny, M. A. Abdelwahab, M. Fakhry, and E. Hasaneen. An efficient vehicle counting method using mask r-cnn. In *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*, pages 232–237, 2020. doi: 10.1109/ITCE48509.2020.9047800.

[241] Mohamed A Abdelwahab. Accurate vehicle counting approach based on deep neural networks. In *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, pages 1–5. IEEE, 2019.

[242] Lu Lou, Qi Zhang, Chunfang Liu, Minlan Sheng, Jun Liu, and Huimin Song. Detecting and counting the moving vehicles using mask r-cnn. In *2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 987–992. IEEE, 2019.

[243] Chenghuan Liu, Du Q Huynh, Yuchao Sun, Mark Reynolds, and Steve Atkinson. A vision-based pipeline for vehicle counting, speed estimation, and classification. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[244] Yomna Youssef and Mohamed Elshenawy. Automatic vehicle counting and tracking in aerial video feeds using cascade region-based convolutional neural networks and feature pyramid networks. *Transportation Research Record*, page 0361198121997833, 2021.

[245] Zhongji Liu, Wei Zhang, Xu Gao, Hao Meng, Xiao Tan, Xiaoxing Zhu, Zhan Xue, Xiaoqing Ye, Hongwu Zhang, Shilei Wen, et al. Robust movement-specific vehicle counting at crowded intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 614–615, 2020.

[246] Zilei Wang, Xu Liu, Jiashi Feng, Jian Yang, and Hongsheng Xi. Compressed-domain highway vehicle counting by spatial and temporal regression. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):263–274, 2017.

[247] Shiva Kamkar and Reza Safabakhsh. Vehicle detection, counting and classification in various conditions. *IET Intelligent Transport Systems*, 10(6):406–413, 2016.

[248] Mohammad Shareef Ghanim and Khaled Shaaban. Estimating turning movements at signalized intersections using artificial neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 20 (5):1828–1836, 2018.

[249] Andres Ospina and Felipe Torres. Countor: Count without bells and whistles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 600–601, 2020.

[250] Shanghang Zhang, Guanhang Wu, Joao P. Costeira, and Jose M. F. Moura. Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[251] Zhe Dai, Huansheng Song, Xuan Wang, Yong Fang, Xu Yun, Zhaoyang Zhang, and Huaiyu Li. Video-based vehicle counting framework. *IEEE Access*, 7:64460–64470, 2019.

[252] Wei Wang, Tim Gee, Jeff Price, and Hairong 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. IEEE, 2015.

[253] Pablo Barcellos, Christiano Bouvié, Fabiano Lopes Escouto, and Jacob 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.

[254] Nam Bui, Hongsuk Yi, and Jiho Cho. A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 578–579, 2020.

[255] Jan Folenta, Jakub Spanhel, Vojtech Bartl, and Adam Herout. Determining vehicle turn counts at multiple intersections by separated vehicle classes using cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 596–597, 2020.

[256] Mohammad Shokrolah Shirazi and Brendan Tran Morris. Trajectory prediction of vehicles turning at intersections using deep neural networks. *Machine Vision and Applications*, 30(6):1097–1109, 2019.

[257] Awad Abdelhalim and Montasir Abbas. Towards real-time traffic movement count and trajectory reconstruction using virtual traffic lanes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 592–593, 2020.

[258] Lijun Yu, Qianyu Feng, Yijun Qian, Wenhe Liu, and Alexander G Hauptmann. Zero-virus: Zero-shot vehicle route understanding system for intelligent transportation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 594–595, 2020.

[259] Mohammad Shokrolah Shirazi and Brendan Morris. A typical video-based framework for counting, behavior and safety analysis at intersections. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1264–1269. IEEE, 2015.

[260] Zhihui Wang, Bing Bai, Yujun Xie, Tengfei Xing, Bineng Zhong, Qinqin Zhou, Yiping Meng, Bin Xu, Zhichao Song, Pengfei Xu, et al. Robust and fast vehicle turn-counts at intersections via an integrated solution from detection, tracking and trajectory modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 610–611, 2020.

[261] Tianyuan Zhang, Xuanyao Chen, Yue Wang, Yilun Wang, and Hang Zhao. Mutr3d: A multi-camera tracking framework via 3d-to-2d queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4537–4546, 2022.

[262] Hao Wu, Xinxiang Zhang, Brett Story, and Dinesh Rajan. Accurate vehicle detection using multi-camera data fusion and machine learning. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3767–3771. IEEE, 2019.

[263] Nimet Kaygusuz, Oscar Mendez, and Richard Bowden. Multi-camera sensor fusion for visual odometry using deep uncertainty estimation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2944–2949. IEEE, 2021.

[264] Apoorv Singh. Transformer-based sensor fusion for autonomous driving: A survey. *arXiv preprint arXiv:2302.11481*, 2023.

[265] Shu Zhang, Elliot Staudt, Tim Faltemier, and Amit K Roy-Chowdhury. A camera network tracking (camnet) dataset and performance baseline. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 365–372. IEEE, 2015.

[266] Minye Wu, Guli Zhang, Ning Bi, Ling Xie, Yuanquan Hu, and Zhiru Shi. Multiview vehicle tracking by graph matching model. In *CVPR Workshops*, pages 29–36, 2019.

[267] Hung-Min Hsu, Jiarui Cai, Yizhou Wang, Jenq-Neng Hwang, and Kwang-Ju Kim. Multi-target multi-camera tracking of vehicles using metadata-aided re-id and trajectory-based camera link model. *IEEE Transactions on Image Processing*, 30:5198–5210, 2021.

[268] Sohaib Khan and Mubarak Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10): 1355–1360, 2003.

[269] Kai-Siang Yang, Yu-Kai Chen, Tsai-Shien Chen, Chih-Ting Liu, and Shao-Yi Chien. Tracklet-refined multi-camera tracking based on balanced cross-domain re-identification for vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3983–3992, 2021.

[270] Hung-Min Hsu, Yizhou Wang, and Jenq-Neng Hwang. Traffic-aware multi-camera tracking of vehicles based on reid and camera link model. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 964–972, 2020.

[271] Andreas Specker, Daniel Stadler, Lucas Florin, and Jurgen Beyerer. An occlusion-aware multi-target multi-camera tracking system. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4173–4182, 2021.

[272] Hung-Min Hsu, Tsung-Wei Huang, Gaoang Wang, Jiarui Cai, Zhichao Lei, and Jenq-Neng Hwang. Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models. In *CVPR Workshops*, pages 416–424, 2019.

[273] Yijun Qian, Lijun Yu, Wenhe Liu, and Alexander G Hauptmann. Electricity: An efficient multi-camera vehicle tracking system for intelligent city. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 588–589, 2020.

[274] Haanju Yoo, Kikyung Kim, Moonsub Byeon, Younghan Jeon, and Jin Young Choi. Online scheme for multiple camera multiple target tracking based on multiple hypothesis tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):454–469, 2016.

[275] Liangjia Zhu, Jenq-Neng Hwang, and Hsu-Yung Cheng. Tracking of multiple objects across multiple cameras with overlapping and non-overlapping views. In *2009 IEEE International Symposium on Circuits and Systems*, pages 1056–1060. IEEE, 2009.

[276] Martijn C Liem and Dariu M Gavrila. Joint multi-person detection and tracking from overlapping cameras. *Computer Vision and Image Understanding*, 128:36–50, 2014.

[277] Elias Strigel, Daniel Meissner, and Klaus Dietmayer. Vehicle detection and tracking at intersections by fusing multiple camera views. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 882–887. IEEE, 2013.

[278] Ran Eshel and Yael Moses. Homography based multiple camera detection and tracking of people in a dense crowd. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[279] Elena Luna, Juan C SanMiguel, José M Martínez, and Marcos Escudero-Viñolo. Online clustering-based multi-camera vehicle tracking in scenarios with overlapping fovs. *Multimedia Tools and Applications*, pages 1–21, 2022.

[280] Hua Tang. Development of a multiple-camera tracking system for accurate traffic performance measurements at intersections, 2013.

[281] Shiloh L Dockstader and A Murat Tekalp. Multiple camera tracking of interacting and occluded human motion. *Proceedings of the IEEE*, 89(10):1441–1455, 2001.

[282] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Haibin Ling, Qinghua Hu, Haotian Wu, Qinqin Nie, Hao Cheng, Chenfeng Liu, et al. Visdrone-vdt2018: The vision meets drone video detection and tracking challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[283] Achal Dave, Tarasha Khurana, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. Tao: A large-scale benchmark for tracking any object. In *European conference on computer vision*, pages 436–454. Springer, 2020.

[284] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[285] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[286] Yunsong Zhou, Yuan He, Hongzi Zhu, Cheng Wang, Hongyang Li, and Qinhong Jiang. Monocular 3d object detection: An extrinsic parameter free approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7556–7566, 2021.

[287] Yu Xiang, Wongun Choi, Yuanqing Lin, and Silvio Savarese. Data-driven 3d voxel patterns for object category recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1903–1911, 2015.

[288] Thibault Buhet, Emilie Wirbel, and Xavier Perrotton. Conditional vehicle trajectories prediction in carla urban environment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[289] Minghan Zhu, Songan Zhang, Yuanxin Zhong, Pingping Lu, Huei Peng, and John Lenneman. Monocular 3d vehicle detection using uncalibrated traffic cameras through homography. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3814–3821. IEEE, 2021.

[290] Hui Miao, Feixiang Lu, Zongdai Liu, Liangjun Zhang, Dinesh Manocha, and Bin Zhou. Robust 2d/3d vehicle parsing in arbitrary camera views for cvis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15631–15640, 2021.

[291] Mengran Gou, Srikrishna Karanam, Wenqian Liu, Octavia Camps, and Richard J Radke. Dukemtmc4reid: A large-scale multi-camera person re-identification dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 10–19, 2017.

[292] James Ferryman and Ali Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE international workshop on performance evaluation of tracking and surveillance*, pages 1–6. IEEE, 2009.

[293] Wenqian Liu, Octavia Camps, and Mario Sznaier. Multi-camera multi-object tracking. *arXiv preprint arXiv:1709.07065*, 2017.

[294] Tatjana Chavdarova and François Fleuret. Deep multi-camera people detection. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 848–853. IEEE, 2017.

[295] Tatjana Chavdarova, Pierre Baqué, Stéphane Bouquet, Andrii Maksai, Cijo Jose, Timur Bagautdinov, Louis Lettry, Pascal Fua, Luc Van Gool, and François Fleuret. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5030–5039, 2018.

[296] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng 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*, pages 8797–8806, 2019.

[297] Sohyeong Kim, Georg Anagnostopoulos, Emmanouil Barmpounakis, and Nikolas Geroliminis. Visual extensions and anomaly detection in the pneuma experiment with a swarm of drones. *Transportation Research Part C: Emerging Technologies*, 147:103966, 2023.

[298] Fabian Herzog, Junpeng Chen, Torben Teepe, Johannes Gilg, Stefan Hörmann, and Gerhard Rigoll. Synthehicle: Multi-vehicle multi-camera tracking in virtual cities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1–11, 2023.

[299] B. Seibold and D.B. Work et al. New directions in mathematical approaches for traffic flow management. Technical report, University of California Los Angeles, 12 2015.

[300] A. Mihály and P. Gáspár. Driver categorization based on vehicle motion and trajectory data. In *2015 16th International Symposium on Computational Intelligence and Informatics (CINTI)*, pages 101–105. IEEE, 2015.

[301] Z. He. Research based on high-fidelity ngsim vehicle trajectory datasets: A review, 2017.

[302] J. Barthélemy, N. Verstaevel, H. Forehead, and P. Perez. Edge-computing video analytics for real-time traffic monitoring in a smart city. *Sensors*, 19(9):2048, 2019.

[303] G. Pettet, S. Sahoo, and A. Dubey. Towards an adaptive multi-modal traffic analytics framework at the edge. In *2019 International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 511–516. IEEE, 2019.

[304] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67, 2017.

[305] X. Fan, G. Fan, and J.P. Havlicek. Generative model for maneuvering target tracking. *Transactions on Aerospace and Electronic Systems*, 46(2):635–655, 2010.

[306] R.E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

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

[308] Tennessee Department of Transportation. Annual Average Daily Traffic (AADT) Maps. Online, accessed December 2022, `https://tdot.ms2soft.com/tcds`, 2022.

[309] Danjue Chen and Soyoung Ahn. Variable speed limit control for severe non-recurrent freeway bottlenecks. *Transportation Research Part C: Emerging Technologies*, 51:210–230, 2015.

[310] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.

[311] Yanbing Wang, Derek Gloudemans, Zi Nean Teoh, Lisa Liu, Gergely Zachár, William Barbour, and Daniel Work. Automatic vehicle trajectory data reconstruction at scale. *arXiv preprint arXiv:2212.07907*, 2022.

[312] Mohammadhossein Bahari, Ismail Nejjar, and Alexandre Alahi. Injecting knowledge in data-driven vehicle trajectory predictors. *Transportation Research Part C: Emerging Technologies*, 128: 103010, 2021. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2021.103010. URL https://www.sciencedirect.com/science/article/pii/S0968090X21000425.

[313] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

[314] Benjamin A. Zielke, Robert L. Bertini, and Martin Treiber. Empirical measurement of freeway oscillation characteristics. *Transportation Research Record: Journal of the Transportation Research Board*, 2088:57–67, 1 2008. ISSN 0361-1981. doi: 10.3141/2088-07. URL http://journals.sagepub.com/doi/10.3141/2088-07.

[315] Martin Treiber, Arne Kesting, and Dirk Helbing. Three-phase traffic theory and two-phase models with a fundamental diagram in the light of empirical stylized facts. *Transportation Research Part B: Methodological*, 44(8-9):983–1000, 2010.

[316] Dirk Helbing, Martin Treiber, Arne Kesting, and Martin Schönhof. Theoretical vs. empirical classification and prediction of congested traffic states. *European Physics Journal B*, 3 2009. doi: 10.1140/epjb/e2009-00140-5. URL http://arxiv.org/abs/0903.0929http://dx.doi.org/10.1140/epjb/e2009-00140-5.

[317] Nathan H Gartner, Carrol Jl Messer, and Ajay Rathi. Traffic flow theory-a state-of-the-art report: revised monograph on traffic flow theory. *Transportation Research International Documentation*, 2002.

[318] Mohammadreza Khajeh Hosseini, Alireza Talebpour, Saipraneeth Devunuri, and Samer H. Hamdar. An unsupervised learning framework for detecting adaptive cruise control operated vehicles in a vehicle trajectory data. *Expert Systems with Applications*, 208:118060, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.118060. URL https://www.sciencedirect.com/science/article/pii/S0957417422012660.

[319] Junping Zhang, Fei-Yue Wang, Kunfeng Wang, Wei-Hua Lin, Xin Xu, and Cheng Chen. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, 2011.

[320] Zi Yang and Lilian SC Pun-Cheng. Vehicle detection in intelligent transportation systems and its applications under varying environments: A review. *Image and Vision Computing*, 69:143–154, 2018.

[321] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[322] S. Chen and C. Shao. Python implementation of the kalman-iou tracker., 2017. https://github.com/siyuanc2/kiout.

[323] Yihong Xu, Yutong Ban, Xavier Alameda-Pineda, and Radu Horaud. Deepmot: A differentiable framework for training multiple object trackers. *arXiv preprint arXiv:1906.06618*, 2019.

[324] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017.

[325] Lester Randolph Ford Jr and Delbert Ray Fulkerson. *Flows in networks*, volume 54. Princeton university press, 2015.

[326] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Liang Zheng, Anuj Sharma, Rama Chellappa, and Pranamesh Chakraborty. The 4th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, page 2665–2674, June 2020.

[327] Katherine F Turnbull. Critical issues in transportation. *TR NEWS*, 2019.

[328] Gerhard P Hancke, Gerhard P Hancke Jr, et al. The role of advanced sensing in smart cities. *Sensors*, 13(1):393–425, 2013.

[329] Ling Hu and Qiang Ni. Iot-driven automated object detection algorithm for urban surveillance systems in smart cities. *IEEE Internet of Things Journal*, 5(2):747–754, 2017.

[330] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[331] Hou-Ning Hu, Yung-Hsu Yang, Tobias Fischer, Trevor Darrell, Fisher Yu, and Min Sun. Monocular quasi-dense 3d object tracking. *arXiv preprint arXiv:2103.07351*, 2021.

[332] Dan Simon and Tien Li Chia. Kalman filtering with state equality constraints. *IEEE transactions on Aerospace and Electronic Systems*, 38(1):128–136, 2002.

[333] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[334] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.01298. URL https://ieeexplore.ieee.org/document/8954311/.

[335] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, Hamburg, Germany, September 2015. IEEE. ISBN 978-1-4799-9994-1. doi: 10.1109/IROS.2015.7353481. URL http://ieeexplore.ieee.org/document/7353481/.

[336] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D Object Detection From RGB-D Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Qi_Frustum_PointNets_for_CVPR_2018_paper.html.

[337] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, Long Beach, CA, USA, June 2019. IEEE. ISBN 978-1-72813-293-8. doi: 10.1109/CVPR.2019.00086. URL https://ieeexplore.ieee.org/document/8954080/.

[338] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10526–10535, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5. doi: 10.1109/CVPR42600.2020.01054. URL https://ieeexplore.ieee.org/document/9157234/.

[339] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337, October 2018. ISSN 1424-8220. doi: 10.3390/s18103337. URL https://www.mdpi.com/1424-8220/18/10/3337. Number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

[340] Nguyen Mau Dung. RTM3D-PyTorch: PyTorch Implementation of the RTM3D paper. https://github.com/maudzung/RTM3D, 2020.

[341] Geonsoo Lee. monocon-pytorch: unofficial pytorch implementation for MonoCon. https://github.com/2gunsu/monocon-pytorch, 2022.

[342] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, 10, 2017.

[343] Collin Castle. Michigan department of transportation cav corridor, 2023. URL https://www.michigan.gov/mdot/travel/mobility/initiatives/cav-corridor.

[344] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[345] Bo Wu and Ram Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 951–958. IEEE, 2006.

[346] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):267–282, 2007.

[347] Sergio M Silva and Cláudio Rosito Jung. A flexible approach for automatic license plate recognition in unconstrained scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5693–5703, 2021.

[348] Gary Bradski. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 25(11):120–123, 2000.

[349] Wireless Estimator. A safe out of plumb monopole is most likely caused by the thermal 'sunflower effect', 2020. https://wirelessestimator.com/articles/2020/a-safe-out-of-plumb-monopole-is-most-likely-caused-by-the-thermal-sunflower-effect/.

[350] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[351] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[352] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[353] Kevin Toohey and Matt Duckham. Trajectory similarity measures. *Sigspatial Special*, 7(1):43–50, 2015.

[354] Libpanda. *Libpanda: A software library and utilities for interfacing with vehicle hardware systems.* The University of Arizona, 2020.

[355] A. Bayen, J. Lee, B. Seibold, B. Piccoli, J. Sprinkle, and D. Work. Circles consortium project website. https://circles-consortium.github.io, 2021.

[356] Yuhang Zhang, Marcos Quinones-Grueiro, William Barbour, Zhiyao Zhang, Joshua Scherer, Gautam Biswas, and Daniel Work. Cooperative multi-agent reinforcement learning for large scale variable speed limit control. In *2023 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 149–156. IEEE, 2023.

[357] Jiaxin Li, Yan Ding, Hua-Liang Wei, Yutong Zhang, and Wenxiang Lin. Simpletrack: Rethinking and improving the jde approach for multi-object tracking. *Sensors*, 22(15):5863, 2022.

[358] Gergely Zachár. Visualization of large-scale trajectory datasets. In *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*, pages 152–157. ACM, 2023.

[359] Marcello Montanino and Vincenzo Punzo. Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns. *Transportation Research Part B: Methodological*, 80: 82–106, 2015.

[360] Benjamin A Coifman and Yun Wang. Average velocity of waves propagating through congested freeway traffic. In *Transportation and Traffic Theory. Flow, Dynamics and Human Interaction. 16th International Symposium on Transportation and Traffic TheoryUniversity of Maryland, College Park*, 2005.

[361] wikipedia. Kalman filter - details, 2023. https://en.wikipedia.org/wiki/Kalman_filter#Details.

[362] Stuart J Russell. *Artificial intelligence a modern approach.* Pearson Education, Inc., 2010.

[363] Stats Stack Exchange. How does one apply kalman smoothing with irregular time steps?, 2022. https://stats.stackexchange.com/questions/49300/how-does-one-apply-kalman-smoothing-with-irregular-time-steps/585962#585962.

[364] wikipedia. Wiener process, 2023. https://en.wikipedia.org/wiki/Wiener_process.

[365] Albert Einstein. *Investigations on the Theory of the Brownian Movement.* Courier Corporation, 1956.

[366] Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[367] Anil Kumar, P Shashidhar Chavan, VK Sharatchandra, Sumam David, Philip Kelly, and Noel E O'Connor. 3d estimation and visualization of motion in a multicamera network for sports. In *2011 Irish Machine Vision and Image Processing Conference*, pages 15–19. IEEE, 2011.

[368] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.