

# Cloud-Based Virtual Traffic Sensor Network with 511 CCTV Traffic Video Streams

FINAL REPORT  
AUGUST 2019

Peter J. Jin  
Assistant Professor

Yi Ge  
PhD Candidate

Terry(Tianya) Zhang  
PhD Candidate

Yizhou Wang  
PhD Candidate

Jonathan Martinez  
Graduate Student

Department of Civil and Environmental Engineering  
Center for Advanced Infrastructure and Transportation (CAIT)  
Rutgers, The State University of New Jersey,  
Piscataway, NJ, 08854

External Project Manager  
Steve Levine, Executive Director  
TRANSCOM

In cooperation with

Rutgers, The State University of New Jersey  
And  
U.S. Department of Transportation  
Federal Highway Administration

## **Disclaimer Statement**

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

The Center for Advanced Infrastructure and Transportation (CAIT) is a National UTC Consortium led by Rutgers, The State University. Members of the consortium are the University of Delaware, Utah State University, Columbia University, New Jersey Institute of Technology, Princeton University, University of Texas at El Paso, Virginia Polytechnic Institute, and University of South Florida. The Center is funded by the U.S. Department of Transportation.

1. Report No. <b>CAIT-UTC-NC56</b>	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle <b>Cloud-Based Virtual Traffic Sensor Network with 511 CCTV Traffic Video Streams</b>		5. Report Date <b>August 2019</b>	
		6. Performing Organization Code <b>CAIT/Rutgers University</b>	
7. Author(s) <b>Peter J. Jin, Yi Ge, Terry(Tianya) Zhang, Yizhou Wang, Jonathan Martinez</b>		8. Performing Organization Report No. <b>CAIT-UTC-NC56</b>	
9. Performing Organization Name and Address <b>Center for Advanced Infrastructure and Transportation (CAIT), Rutgers, The State University of New Jersey, Piscataway, NJ, 08854</b>		10. Work Unit No.	
		11. Contract or Grant No. <b>DTRT13-G-UTC28</b>	
12. Sponsoring Agency Name and Address <b>Center for Advanced Infrastructure and Transportation Rutgers, The State University of New Jersey 100 Brett Road Piscataway, NJ 08854</b>		13. Type of Report and Period Covered <b>Final Report 9/1/2018-9/30/2019</b>	
		14. Sponsoring Agency Code	
15. Supplementary Notes <b>U.S. Department of Transportation/OST-R 1200 New Jersey Avenue, SE Washington, DC 20590-0001</b>			
16. Abstract <p>The existing 511 NJ traffic data feeds provide real-time travel time and event data. However, travel time is only one of the characteristics of traffic flow and the lack of flow and occupancy information make it difficult for transportation agencies to apply advanced TSM&amp;O (Transportation System Management &amp; Operations) strategies that need comprehensive inputs including travel time (speed), flow, and occupancy. This project developed a cloud-computing platform for state-wide traffic video data analytics. The platform will use the existing 511 CCTV traffic video streams to generate traffic flow and occupancy data to enrich the existing TRANSCOM data feed primarily containing travel time and event data.</p> <p>A longitudinal scanline based algorithm is developed to detect vehicle trajectories with minimal computational cost. Special treatment modules including static noise removal, clustering removal, and strand detection methods are developed to address the occlusions and lane-changing problems in the NJ511 CCTV traffic videos. The algorithm is deployed in an Amazon Web Service (AWS) based platform for large-scale processing of the large number of video feeds. The research outcome includes traffic flow data matched with TRANSCOM TransFusion data links. The evaluation results indicate promising accuracy, computational efficiency, and low cost of the proposed platform.</p>			
17. Key Words <b>Traffic Flow, Video Analytics, Cloud Computing, Traffic monitoring</b>		18. Distribution Statement	
19. Security Classification (of this report) <b>Unclassified</b>	20. Security Classification (of this page) <b>Unclassified</b>	21. No. of Pages <b>Total # 60</b>	22. Price

## **Acknowledgments**

The project team would like to thank TRANSCOM and NJDOT for agreeing to review the project outcome and their suggestions and insights provided to the project team.

## Table of Contents

1.	Introduction and Problem Definition .....	1
2.	Literature Review.....	2
2.1.	Overview of Video Analytic Technologies .....	2
2.1.1.	Integrated camera and analytic solutions .....	2
2.1.2.	Universal virtual sensor solutions .....	3
2.1.3.	Cloud-based smart city video analytic solutions.....	6
2.2.	Overview of Video Analytic Algorithms .....	6
2.2.1.	Video-based Vehicle Trajectory Extraction Methods.....	6
2.2.2.	Video Analytics with the Scanline-based Spatial-Temporal Diagrams .....	7
3.	System Design and Architecture.....	9
4.	Video Analytic Models .....	10
4.1.	Video Characteristics used in Analytic Algorithms .....	10
4.2.	The Proposed STLine-based Video Analytic Algorithms .....	11
4.3.	STLine Marking .....	14
4.4.	STLine-direction-based Camera and Lane Direction Determination .....	14
4.5.	Static Noise Removal on STMap .....	17
4.6.	STMap-based Vehicle Crossing Removal.....	18
4.7.	STMap-based Vehicle Occlusion Detection and Separation.....	18
4.8.	STMap-based Vehicle Counting Combined with Lane-changing Detection .....	20
5.	Cloud computing architecture and system configuration.....	20
5.1.	Daily Reboot.....	21
5.2.	Daily Runner .....	21
5.3.	Python Configuration .....	22
6.	TransFusion Link Data Processing .....	22
6.1.	ArcGIS-based Direction Processing.....	23
6.1.1.	Feature to line Tool .....	23
6.1.2.	Create New Network Dataset.....	23
6.1.3.	Create New Route .....	24
6.1.4.	Export Directions of Routes .....	24
6.2.	Matching between TransFusion Links and CCTV Cameras .....	25
7.	Traffic Database Management System .....	25
7.1.	Database Management.....	25

8.	System Calibration and Evaluation.....	28
8.1.	Model Evaluation .....	28
8.2.	Key Parameters.....	28
9.	System Evaluation and Pilot Testing Results.....	30
9.1.	CCTV Traffic Camera Data Sources.....	30
9.2.	Video Analytic Model Validation Results.....	31
9.2.1.	ST-Line Creation and ST Map Generation and Denoising .....	31
9.2.2.	Strand Clustering Processing Results.....	32
9.3.	Lane Direction Results .....	36
9.4.	Traffic Flow Detection Results.....	37
9.5.	Cloud Deployment and Computational Cost Estimation.....	39
9.6.	Limitations of the Proposed Models.....	40
9.6.1.	Lane direction determination for Type 4 & Type 7 .....	40
9.6.2.	Streak caused missing information in STMaps.....	41
9.6.3.	vehicle coverage caused by low angle .....	42
9.6.4.	PTZ operations caused STLine matching issue .....	43
10.	Cloud Deployment Strategies and Cost Analysis Results.....	43
10.1.	Proposed System Deployment Schematics with Existing TRANSCOM Systems.....	43
11.	Conclusions and Future Work .....	47
12.	Appendix.....	48

## List of Figures

Figure 1 CitiLog Video Analytic Applications.....	3
Figure 2 Sample Snapshots of Traffic Vision Products.....	4
Figure 3 MetroTech Traffic Video Analytic Applications .....	4
Figure 4 Sample Outputs from the GoodVision Traffic Counting System.....	5
Figure 5 User Interface of GoodVision Video Analytic System .....	6
Figure 6 The Proposed System Framework for Cloud-based Video Traffic Counter .....	9
Figure 7 The Generation of STMap (Spatial-Temporal Map) From CCTV Video Input.....	10
Figure 8 Examples of STLines in Camera View .....	14
Figure 9 STLine-the connection between Camera View and STMap for Camera Direction Determination .....	15
Figure 10 Illustration of the Static Noise (e.g. Light poles, lane markings, etc.) Removal Algorithms.....	17
Figure 11 Illustration of the Proposed Occlusion Detection and Removal Algorithm .....	19
Figure 12 The Proposed Cloud Computing Architecture and System Configuration.....	20
Figure 13 Create New Network Dataset for Matching with the TRANSCOM Link System.....	23
Figure 14 Create New Route for Geospatial Matching with CCTV Traffic Camera Locations.....	24
Figure 15 Export Directions of Routes for Geospatial Matching with CCTV Traffic Camera Locations ..	24
Figure 16 Final Direction Results Generated for Geospatial Matching with Camera Locations.....	25
Figure 17 Matching Example between the Cameras and its Adjacent Links.....	25
Figure 18 Database Entity-Relationship Diagram (PK: Primary Key; FK: Foreign Key) for Database Tables used in the Proposed Platform .....	26
Figure 19 Camera View and STLines.....	30
Figure 20 Snapshot of the VLC Traffic Counter for Generating Ground Truth Data.....	30
Figure 21 STMap Generation from Traffic Video with Pre-marked STLines.....	31
Figure 22 Sample output of Canny Edge Processing Algorithms for the STMap .....	32
Figure 23 Sample Output of Time Differencing Processing of the STMap.....	33
Figure 24 Sample Output of Thresholding .....	34
Figure 25 Sample Output of Canny Combined with Time Difference and Threshold .....	34
Figure 26 Sample Output of Filled Canny .....	35
Figure 27 Sample Output after Removing Noise by Duration.....	35
Figure 28 Sample Output of Counted Strands .....	35
Figure 29 Lane Direction Determination Sample .....	36
Figure 30 Count Results.....	37
Figure 31 Strand Detection Sample .....	39
Figure 32 CPU Occupancy Sample .....	39
Figure 33 Performance of Deployment Tests on Sep. 6th, 2019 .....	40
Figure 34 Lane Direction Determination Sample at US 1 at Bakers Basin Rd. ....	40
Figure 35 Streak Samples at 4th min in the tested 12-min video.....	41
Figure 36 Sample of Coverage by Large Vehicles in Neighboring Lanes.....	42
Figure 37 Sample of Coverage by Vehicles in Same Lane.....	42
Figure 38 Sample of STLine Shifting Caused by PTZ Operations.....	43
Figure 39 Private Cloud Deployment Schematics with Existing TRANSCOM Systems .....	44
Figure 40 Amazon Cloud Deployment Schematics with Existing TRANSCOM Systems .....	45

## List of Tables

Table 1 Key modules and sample outputs of the proposed video analytic algorithms .....	11
Table 2 Relations between Camera Directions and STLine Pixel Coordinate Characteristics.....	15
Table 3 Crontab Scheduling Parameters Configured in Amazon Web Services(AWS).....	21
Table 4 Python Modules Deployed in AWS Cloud Computing Platform .....	22
Table 5 Selected Links used for the Evaluation of the Proposed Platform.....	22
Table 6 Data Fields in TMC_IDENTIFICATION_5K Table.....	26
Table 7 Data Fields in CAMERA_POSITION Table.....	27
Table 8 Data Fields in TMC_CAMERA_MATCH Table.....	27
Table 9 Data Fields in ST_LINE Table .....	27
Table 10 STLines Data Structures .....	27
Table 11 Data Fields in T_NJCloudCounter_MOES Table .....	28
Table 12 Key parameters .....	29
Table 13 Comparison between Commercial Cloud Deployment and Personal Cloud Deployment.....	46



## **1.Introduction and Problem Definition**

The CCTV (Closed Circuit Television) traffic surveillance systems are one of the most important assess of traffic management centers (TMCs) to monitor congestion and incidents in daily operations. Computer vision sensors based on CCTV traffic cameras have been explored over the last few decades. Most earlier commercialized systems rely on software programs or hardware computing units fully calibrated for individual CCTV traffic cameras often preconfigured to fixed angles (Michalopoulos, 1991). Recently, researchers and engineers started to develop easy-to-calibrate computer vision systems for PTZ(Pan-Tilt-Zoom) cameras (Song&Tai, 2016; Birchfield et al, 2010). Such systems can take advantage of the existing large number of CCTV traffic cameras operated by TMCs to generate speed, flow, occupancy or even trajectory data to support not only the existing TSM&O (Transportation Systems Management and Operations) and also the emerging Connected and Automated Vehicle (CAV) applications. The rapid growth of cloud computing and crowdsourcing technologies provides new opportunities for deploying such high-resolution computer vision systems at the state or regional level for large-scale traffic operations and CAV system needs.

Current commercialized systems can be classified into two major categories, the integrated hardware-software solutions, and the generic computer vision solutions. The former focuses on building customized video analytic models fully calibrated with a specific type of camera. Representative products include Autoscope (1984), Gridsmart (2006), etc. The integrated nature of such systems can ensure the software analyzes the full resolution of the video data at the “edge” while achieving optimal results with fully calibrated models. However, deploying such systems will need cameras installed at dedicated locations and the cameras cannot be used for other surveillance purposes. The latter approach emerges with the rapid development of cloud computing technologies and computer vision technologies. The new generation of computer vision systems does not rely on tight integration with specific camera models. The deep learning and latest computer vision models can be used by process traffic video from any scenes with little or no manual input information such as scanlines, detection zones. Representative solutions include CitiLog (1997), TrafficVision (1999), MetroTech (2012), and Good Vision. These solutions can be deployed to existing CCTV video systems without the need for additional hardware installation. However, due to the complexity of the computer vision algorithms used, most of the systems can only process the data offline to achieve high accuracy.

In New Jersey, the main real-time transportation data categories include travel time and traffic event data. Such data are provided through the live data feed from TRANSCOM, a coalition of more than 19 transportation agencies in the tri-state area. TRANSCOM travel time data comes from probe vehicle data collected through EZ-Pass readers (including those for non-toll purposes), Inrix, and HERE(Nokia) probe vehicle travel time data. The EZ-Pass reader data are published to the public; while the full TRANSFusion data integrating all three data sources are only for member transportation agencies.

Nevertheless, the lack of real-time traffic flow data in the dynamic data feed offerings has led to some severe limitations in traffic operations and control. Without the flow data, it is difficult to tell whether or not a free-flow segment is in factor free-flow or closed. It is also difficult to tell whether or not the traffic on a free-flow travel time segment has actually light traffic or saturated flow that may break down at any time. Many regional freeway and arterial traffic control solutions, such as Active Traffic Management (ATM) and adaptive traffic signal control. The coverage of fixed-location detectors such as loop detectors, RTMS (remote traffic microwave sensor), traffic dots or pucks are still limited and not connected to any real-time dynamic traffic data feed. Meanwhile, the more than 400 CCTV traffic cameras deployed throughout the state and major roadways in NJ can potentially form a large virtual sensor network to generate traffic flow data to be incorporated into the dynamic traffic flow data feed.

This study focuses on the development of an efficient cloud-based online video analytic system for generating traffic flow data from large-scale regional CCTV traffic video network. The proposed method

can be classified into the generic video analytic solutions since the proposed computer vision algorithms are not designed for specific types of video cameras and can be applied to the video streams from the existing CCTV traffic surveillance cameras. The proposed video analytic algorithms can address the issues including computational efficiency for online deployment, the automated lane direction determination due to PTZ operations, occlusions in multi-lane traffic video, and lane changes.

The proposed solution has several key modules. First, the camera location and direction determination module which determines the road name and directions of the lane-by-lane traffic detected by the proposed method. Second, the video analytic algorithms that improve an existing model (Zhang, 2019) to generate lane-by-lane traffic flow. Third, the cloud-based parallel computing platform that uses AWS (Amazon Web Services) cloud to simultaneously calculate multiple CCTV traffic video streams at the same time. Fourth, mapping and reporting modules that map the generate lane-by-lane flow data with TRANSFusion link systems.

## **2.Literature Review**

### **2.1.Overview of Video Analytic Technologies**

Existing traffic video analytic systems can be classified into three major categories including integrated camera and analytic solutions, universal virtual sensor solutions, and cloud-based smart city video analytic solutions. Representative products include Autoscope (1984), Citilog (1997), VISATRAM (Zhu, 2000), and GRIDSMART (2006) systems. The analytic software in those systems often highly customized to fit the features of their OEM cameras. OEM cameras also reduce the need for frequent camera calibration. The key limitation is the closed system making it difficult to integrate other existing agency traffic video resources, and the pricing can be non-competitive with dedicated systems.

#### **2.1.1.Integrated camera and analytic solutions**

In the first category, image processing algorithms are developed and fully-calibrated for specific types of video cameras. Many of those systems even encode the image processing algorithms within the processing units directly connected to the cameras. Such tight integration allows those algorithms to take full advantage of the full resolution and quality of the raw video to generate needed traffic data. Representative platforms include Autoscope (1984) and Gridsmart (2006).

- **Autoscope Systems**

Image Sensing Systems, Inc. emerged in 1984 which focused on developing and delivering above-ground detection technology, applications, and solutions. Image Sensing Systems combined video, radar, Bluetooth for detection and supplied wrong way detection solutions and IntellitracfiQ solutions. Now, it has more than 140,000 Autoscope Vision units sold in over 70 countries worldwide.

- **Gridsmart Systems**

GRIDSMART Technologies Inc. was founded in 2006. The company pioneered the world's first single-camera solution for intersection actuation, traffic data collection, and situational awareness. GRIDSMART uses one Omni-vision camera to monitor an entire intersection. GRIDSMART used a single camera with appropriate video processing to monitor the whole intersection, which has counted and classified more than 216 billion vehicles. The fisheye-camera-based Iconic GRIDSMART Bell Camera have been installed in 1200 cities, 49 states, and 29 countries.

The limitations of the integrated hardware-software platforms are their dependency on the installation and configuration of the hardware systems. Many transportation agencies many already have their surveillance video systems but in order to use integrated hardware-software platforms. The agencies need to install new dedicated fixed-position fixed-view video cameras for those platforms to generate data.

### 2.1.2. Universal virtual sensor solutions

Universal virtual sensor platforms are essentially hardware independent and developed for typical intersection or roadside traffic scenes often observed in existing CCTV traffic cameras. Some platforms can be deployed to existing traffic data sources even PTZ cameras (e.g., TrafficVision (1999)) with the capability of adjusting the “scanlines” or “detector zones” in PTZ operations. Representative systems include CitiLog (1997), TrafficVision (1999), MetroTech (2012), GoodVision (2017), Miovision (2005), KiwiVision (2011), Aventura (1999). However, significant manual work is still needed to set up virtual detection zones for each camera position and can be difficult for on-demand video sources analytics on-the-fly. The output is simple traffic states derived from vehicle occurrence detection in the manually-drawn detection zone. The occurrence-based detection limits the resolution of the data to support more complicated measures, e.g., advance detection for Purdue diagram analysis. Recently, with the development of AI(Artificial Intelligence) technologies, especially, deep learning models such as Mask RCNN (Region-based Convolutional Neural Networks) (Bharati, 2019; Omeroglu, 2019) and YOLO (Corovic, 2018; Iwasaki, 2018; Lin, 2018), the accuracy of some latest generic computer vision platforms has been significantly improved over the years. Though running those AI models will require significant computational resources and is often done through cloud computing services and cannot be achieved in real-time. Representative video analytic systems are as follows.

- **CitiLog**

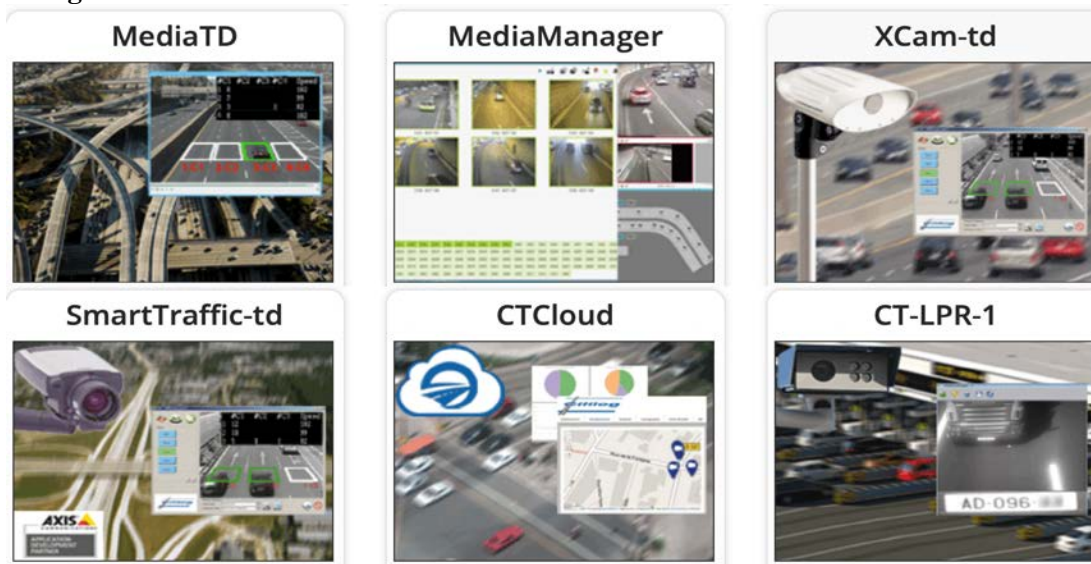


Figure 1 CitiLog Video Analytic Applications

CitiLog was founded in 1997. CitiLog has both integrated hardware and video analytic technologies with its AVIX IP cameras (<http://www.citilog.com/product/en/smartcam>). The company also has universal detection technologies that can be applied to existing cameras. As Figure 1 shows, CitiLog's traffic detection suites can be used to detect traffic parameters (speed, flow, density) such as MediaTunnel, MediaRoad, MediaTD, MediaManager, intersection flow and queue lengths such as XCam-p, XCam-ng, XCom, SmartTraffic-p, SmartTraffic-ng, incidents such as VisioPaD, VisioPaD+, SmartTraffic-AID, SmartTraffic-i, SmartTraffic-ww, license plates for generating toll, tracking, and journey travel time information such as CT-LPR-1, CT-LPR-2, CT-HAZ. Its products monitor over 900 sites in 55 countries and have been processing 32,000 video inputs every day.

- **TrafficVision**

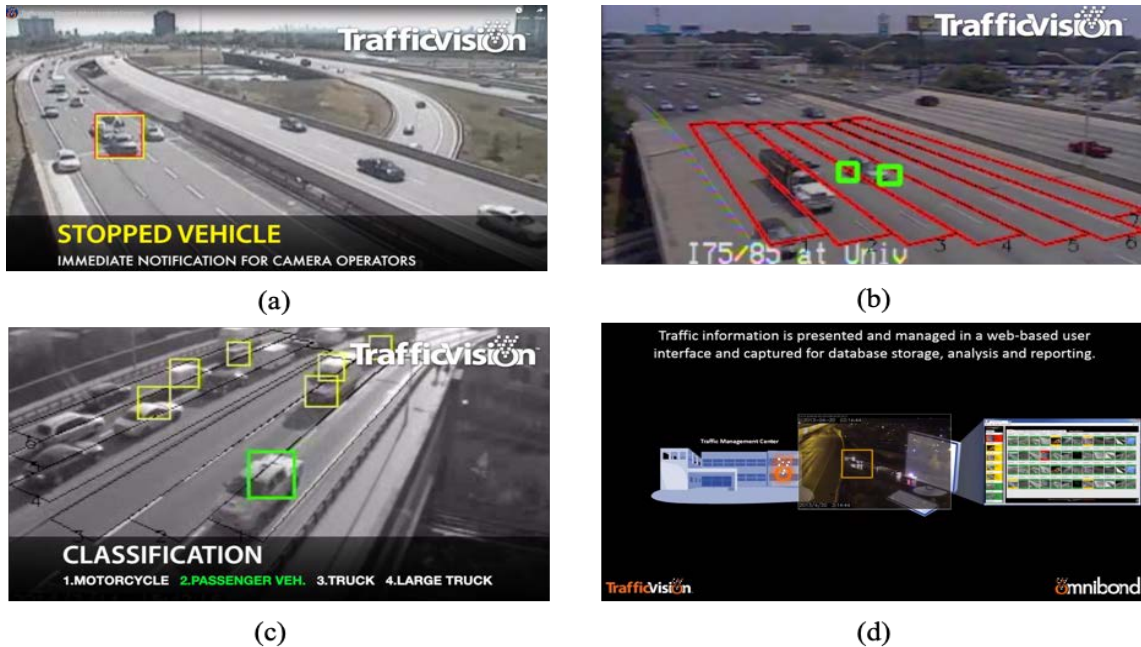


Figure 2 Sample Snapshots of Traffic Vision Products

TrafficVision is a division of Omnibond (<http://www.trafficvision.com>) founded in 1999. It has been focused on real-time traffic monitoring and has high versatility on data collection and incident detection. The company supplies applications for traffic detection such as Incident Detection (Stopped vehicle, wrong way, slowed traffic), Automatic Re-Calibration, Vehicle Classification. To be mentioned, its real-time incident detection enables immediate alert. Now TrafficVision has helped Ministry of Transportation Ontario collect over 20,000 hours of traffic data around Toronto.

- **MetroTech**



Figure 3 MetroTech Traffic Video Analytic Applications

MetroTech is a company that earned its fame since 2012 (<https://metrotech-net.com>). It provides precise, cost-effective and real-time information by aggregating existing sensors and information. The company supplies MetroTech Family of Products, such as IntelliSegment for piece data collection, IntelliSection for data analysis and distribution, MetroTech Digital Streets Fusion Center for data aggregation and the MetroTraffic Network for global data production. MetroTech can apply their video analytic models to existing traffic cameras with some manual setup of the detection zones. MetroTech was recognized as an up and coming competitor in the industry of smart infrastructure by Government Technology (<https://www.govtech.com/100/2019>).

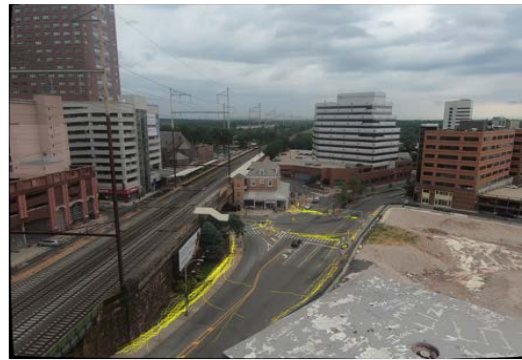
- **GoodVision**

GoodVision (2017) is an online deep-learning-based traffic video analytics tool with a price of €15 per video-hour. It has a comprehensive web interface for uploading video and analyzing the results as shown in the figure below.

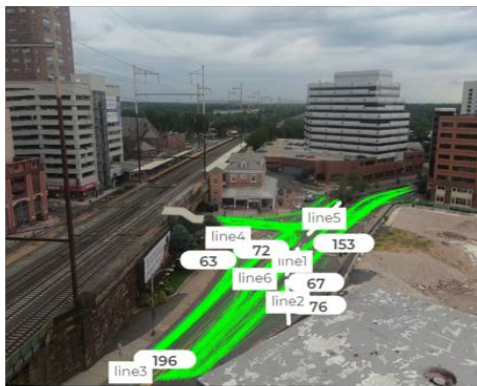
It is good at dealing with high-resolution traffic video at a middle angle or high angle and can not only classify the moving objects into trucks, vans, motorbikes, pedestrians, etc. but also display the graphical results. It does not require the customers to install any software, but it asks the customers to upload recorded videos and wait patiently for the results. The GoodVision application can also output customized traffic count results by drawing virtual zones or stop-bars on the web interface as shown in the figure below. It also has the limitations of distance from monitored objects, obstacles, lense quality, lighting conditions, and image resolutions.



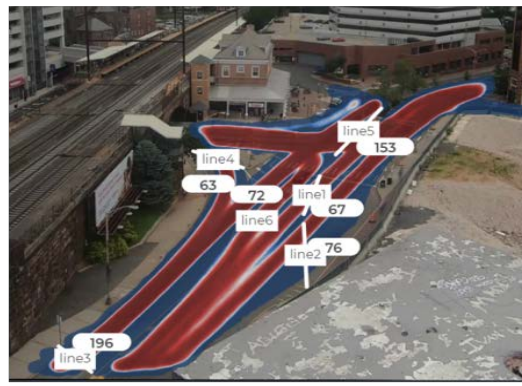
(a) Vehicle Trajectory Detection from GoodVision



(b) Pedestrian Trajectory Detection from GoodVision



(c) Virtual StopBar Counter from GoodVision



(d) Detection Frequency Report from GoodVision System

Figure 4 Sample Outputs from the GoodVision Traffic Counting System

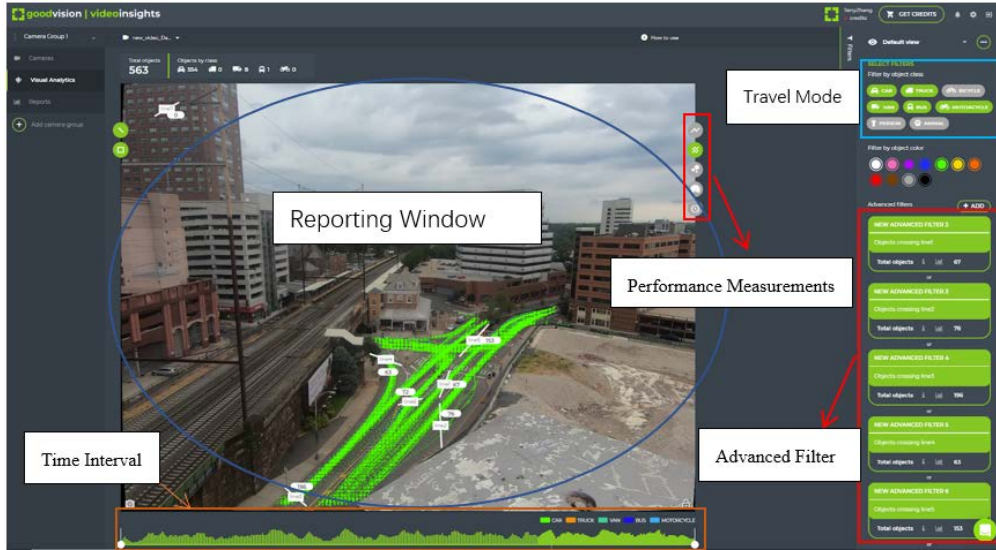


Figure 5 User Interface of GoodVision Video Analytic System

Figure 5 shows the GoodVision application user interface to generate traffic detection and performance metrics. The reporting panel demonstrates the detected trajectory results. Travel mode panel allows users to select different travel modes. The advanced filter can be used to select zone-based or stop-bar based outputs. The time interval panel can be used to select a specific time duration.

Those systems can be deployed to existing traffic data sources even PTZ cameras (e.g., TrafficVision (1999)) and are flexible with existing agency traffic video sources. However, significant manual work is still needed to set up virtual detection zones for each camera position and can be difficult for on-demand video sources analytics on-the-fly. The output is simple traffic states derived from vehicle occurrence detection in the manually-drawn detection zone. The occurrence-based detection limits the resolution of the data to support more complicated measures, e.g., advance detection for Purdue diagram analysis.

### 2.1.3. Cloud-based smart city video analytic solutions

The advance in central (CPU) and graphics processing unit (GPU) technologies and cloud computing triggers the current wave of cloud-based smart city video analytic solutions. Many platforms such as Placemeter (2012), MicroFocus (ex-HP), IBM Intelligent Video Analytics, Cisco Smart City Cloud, BriefCam, have the capability of crowdsourcing and on-demand video analytics. However, those systems often offer heavy-weight all-inclusive packages of smart city functionalities such as TSM&O (Transportation System Management and Operations), energy and utility management, etc. There have not been light-weight cloud-based crowdsourcing video analytic solutions dedicated to traffic state detection in the market.

## 2.2. Overview of Video Analytic Algorithms

### 2.2.1. Video-based Vehicle Trajectory Extraction Methods

To overcome the inadequacy of trajectory data, many researchers have attempted to develop computer vision algorithms to collect traffic data in the last two decades. The vehicle detection and tracking are the two main steps of traffic video analysis. The detection step separates the object of interest from the background and then recognizes the location and scale of the targeted object. The tracking step identifies the object of interest in consecutive frames to trace object movements.

The vehicle detection algorithms can be divided into two major categories: the model-based method and the motion-based method. Model-based methods can localize and classify the objects of interest based on their shapes and understand the context in the traffic scene. For instance, they can classify vehicles into

different types including the bus, car, truck, motorbike, etc. Traditional model-based methods apply sliding windows of different sizes to search around the image and examine if there is an object in it (Felzenszwalb et al., 2010; Ke et al., 2018). Later on, the sliding window method is replaced with the Region Proposal Convolutional Neural Network (R-CNN), where a CNN model will propose the regions of interest (ROI) (R-CNN, Girshick et al., 2015; Faster-CNN, Ren et al., 2017). More recently, novel methods were proposed to detect multiple objects using a single neural network without region-proposal stages, such as the SSD (Single Shot MultiBox Detection, Liu et al., 2016) and YOLO (You Only Look Once, Redmon et al., 2016). However, many of these model-based methods, especially deep neural networks, need large training datasets, complicated design of deep convolutional network structures, and long training times to achieve optimal performance for targeted scenes.

In contrast to the model-based method, motion-based approaches use the frame differencing, optical flow, and other motion-related information to segment out moving blobs. Background subtraction, optical flow, and frame difference are three commonly seen motion-based algorithms (Dailey et al., 2000; Pumrin and Dailey, 2007). Background subtraction is considered a key approach in intelligent surveillance systems with fixed cameras (Stauffer and Grimson, 1999; Kaewtrakulpong and Bowden, 2001; Magee, 2004). The limitations of the background subtraction approach are the background updating strategies and adjustment against illumination changes. Optical flow allows us to extract motions from video streams based on the displacements of vector fields between two consecutive frames (Barron et al., 1992). Although the optical flow method can capture the subtle pixel movements, it is restricted to the constant brightness assumption, resulting in the incomplete detection of moving objects.

For the tracking stage, a major problem is the loss of accuracy over a long duration under the rapidly-changing environment due to the illumination, weather, glare, shadow, and other scenery changes. Traffic surveillance video involves frequent occlusions, various appearances of vehicles, and intensive interactions between the leading and following cars. Compared with tracking a single object, the task for tracking multiple objects imposes more difficulties. A popular technique for vehicle tracking is the Kalman Filter (KF), which is considered an efficient approach for estimating the dynamic state of a system for tracking (Rad and Jamzad, 2005; Hsieh et al., 2006; Kim and Cao, 2010). Many researchers have tried to add some prior knowledge, such as movement constraints, locations, or past tracking histories to better approximate the object locations (Shi and Thomasi, 1994; Coifman et al., 1998; Azevedo et al., 2014). However, the tracked objects can still drift over time due to the error accumulation from the first to the last frame. Deep Convolutional-Neural-Network-based trackers can potentially achieve higher accuracy. However, it needs a large amount of training data, for example, vehicle pixel models for cars, trucks, buses, etc. with different types, colors, models, and under different illumination and weather conditions. The significant manual-processing efforts in preparing the training datasets and the needed computational resources (e.g., GPUs (Graphics Processing Units) based cloud or parallel computing) made those models less computationally- and resource-efficient for the targeted high-angle traffic video analysis applications.

The above limitations call for more computationally-efficient, accurate, and robust models that can be used to detect vehicle trajectories from the high-angle traffic monitoring video.

### **2.2.2. Video Analytics with the Scanline-based Spatial-Temporal Diagrams**

In the proposed research, we focus on one category of video analytics methods which are named as scanline-based methods. Such methods only process a very small portion (pixels on the scan-lines) of the entire video image, with low computational cost and less sensibility towards illumination changes. There are two types of scanlines studied in existing literature, the lateral and the longitudinal scanlines. The lateral scanline is a cross-section scanline across a lane; while the longitudinal scanline is a line along the direction of traffic.

The lateral scanlines can be used to detect flow, vehicle class and spot speed (e.g., with two lateral scanlines together) effectively. Tseng et al. (2002) used the virtual lateral lines across the highway shown in traffic videos to detect, classify, and track vehicles. Zhang et al. (2007) adopted a similar lateral scan-line concept

to detect vehicles by comparing the pixel values along a detection line between the composed background image and the current input frame. Mithun et al. (2012) introduced the concepts of the virtual detection line (VDLs) for vehicle detection and classification. Ren et al. (2014) further explored virtual detection lines (VDLs) to extract vehicle count, mean speed, and vehicle types from multiple foreground temporal-spatial images (TSIs).

The longitudinal scanlines can be used to track vehicles traveling within a lane and preserve the vehicle trajectories. Zhu et al. (1996) have invented a system called VISTRAM using both lateral and longitudinal scanlines to count vehicles, estimate speeds and classify vehicle types. Taniguchi et al. (1999) implemented a longitudinal scanline method to estimate traffic parameters, which was named as the directional temporal plane transformation (DTT). Cho and Rice (2007) investigated a longitudinal mask-based method to estimate traffic parameters on I-80 in California, and a shifting and matching algorithm was applied to estimate a weighted median velocity over a region of the space and time. Malinovskiy et al. (2009) proposed an improved and upgraded scanline-based methodology for vehicle detection and most importantly vehicle tracking using ST (Spatial-Temporal) maps with CCTV cameras on highway segments. In their paper, they applied the Hough Transformation to analyze the strands and grouped Hough lines based on geometry properties. The limitation of Hough Line detection is that it only works when the vehicle trajectory on the ST Map is straight. The main outputs of the proposed algorithm are primarily traffic flow counts. Ardestani et al. (2016) applied the longitudinal scanline-based algorithm to detect traffic signals from regular CCTV cameras available at major arterial intersections. By analyzing the patterns of queuing vehicles in the ST Map, the algorithm can identify the starting and ending time of signal phases efficiently. In this paper, rather than just detecting aggregated traffic states, the objective is to generate real-world vehicle trajectories that are comparable to NGSIM data.

Another important aspect of traffic video analysis is to reconstruct real-world trajectories through camera calibration. Schoepflin and Dailey (2003) introduced a three-stage dynamic camera calibration process for roadside traffic cameras. Cathey and Dailey (2005) have proposed a technique to remove the perspective effects from CCTV cameras based on 2D/3D motion models. The related formula of the coordinate transformation and the solution to obtain the transformation matrix will be provided in the methodology section.



### 3. System Design and Architecture

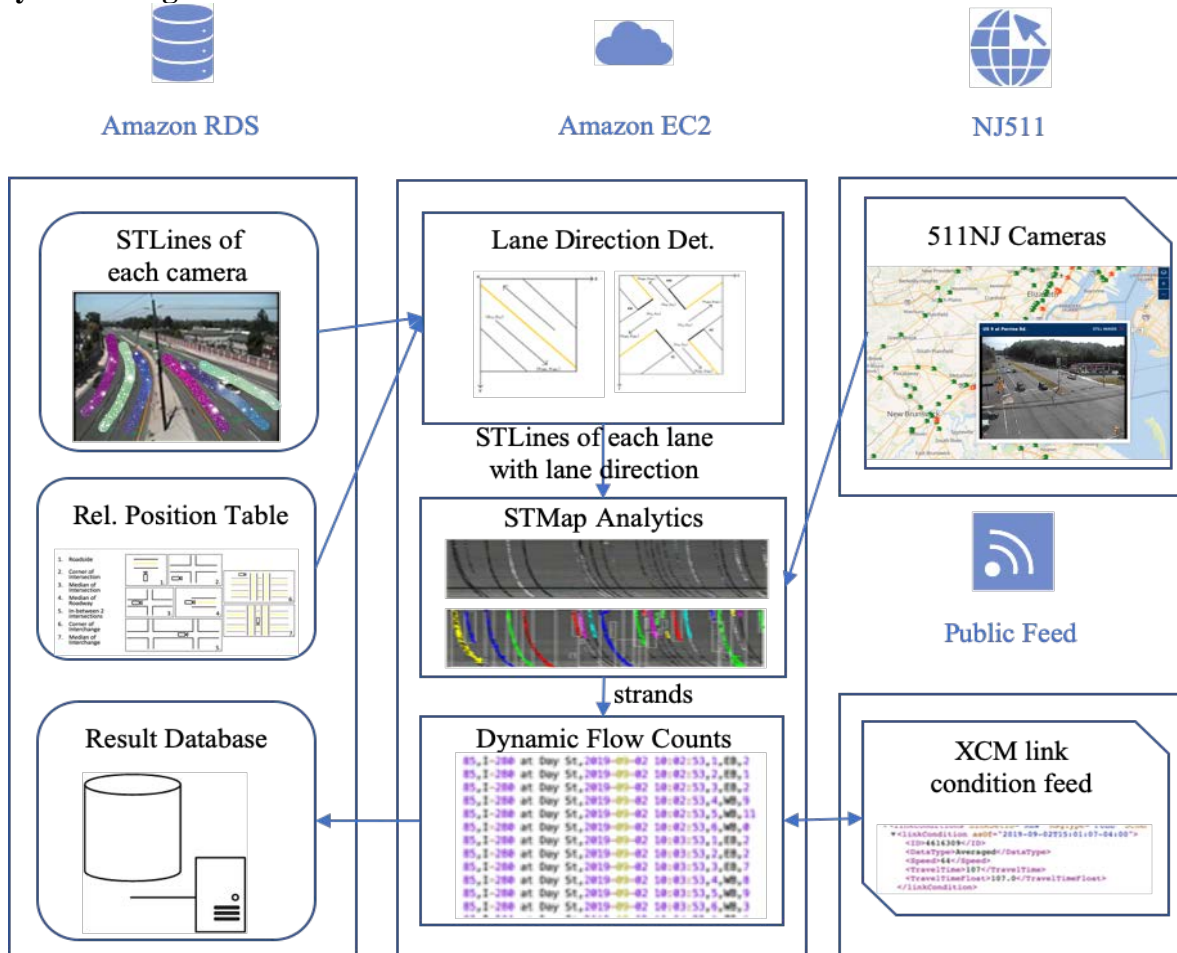


Figure 6 The Proposed System Framework for Cloud-based Video Traffic Counter

Figure 6 shows the design of the proposed cloud-based traffic counter system based on CCTV traffic cameras. The system will be built on two cloud platforms including a cloud database system (Amazon RDS (Relational Database System)) and a cloud computing system (EC2 (Elastic Compute Cloud)). The system takes the traffic video feed from 511 NJ cameras, generate dynamic traffic flow data that can be added to TRANSCOM (XCM) link condition data feed. Several key system components are summarized as follows.

- STLine Generation:** The STLines are longitudinal scanlines with which the video analytic algorithms detect and track vehicles. The STLines will be manually processed for all CCTV traffic camera feeds used and need to be periodically updated to reflect changes due to major PTZ operations. The automated algorithm will be developed to readjust the STLines due to minor PTZ operations. The output of the STLine generation modules is detailed lane-by-lane ST scanline geometries with all pixel coordinates of the turning points recorded. The results will be fed into the lane direction determination module and the STMap analytic module.
- Lane Direction Determination:** The relative positions of cameras are critical in the proposed cloud counter since it is the key information for lane direction detection, which can match the generated flow count with its corresponding direction of the traffic flow. A table containing the camera installation information can be acquired from NJDOT. The output of this module will be used in the lane direction determination module.
- STMap Analytics:** This is the core video analytic module that generates and analyzes the STMap (Spatial-Temporal Map) from traffic video streams. Every STMap consists of accumulated pixels from

the scanline of each lane from continuous video frames. STMap turns the 3Dimension video\*time into 2Dimension line\*time. Some denoising techniques are introduced to remove the impact of static objects, occlusions, and lane changes. Traffic counts are conducted by tracking the strands generated by vehicles on the STMap. The output of the STMap analytics is vehicle counts for the Dynamic Flow Counting module.

- **Dynamic Flow Counting:** The vehicles can be counted based on the extracted strands from the STMap Analytics module. The detected strands are vehicle parts extracted by the corresponding STLine, one strand for one vehicle. The counts of this module will be stored in Amazon RDS and integrated with the XCM feed.
- **Result Archiving, Exporting, and Integration with XCM Feed:** The result generated by dynamic flow counting will be stored in the database for long-term storage and other analysis in the future. Meanwhile, the result will also be integrated with XCM feed and published to the public.

#### 4.Video Analytic Models

##### 4.1.Video Characteristics used in Analytic Algorithms

The proposed algorithm is one special characteristic of traffic video, the spatial-temporal map. The detailed definition of the scanline and STMap is as follows.

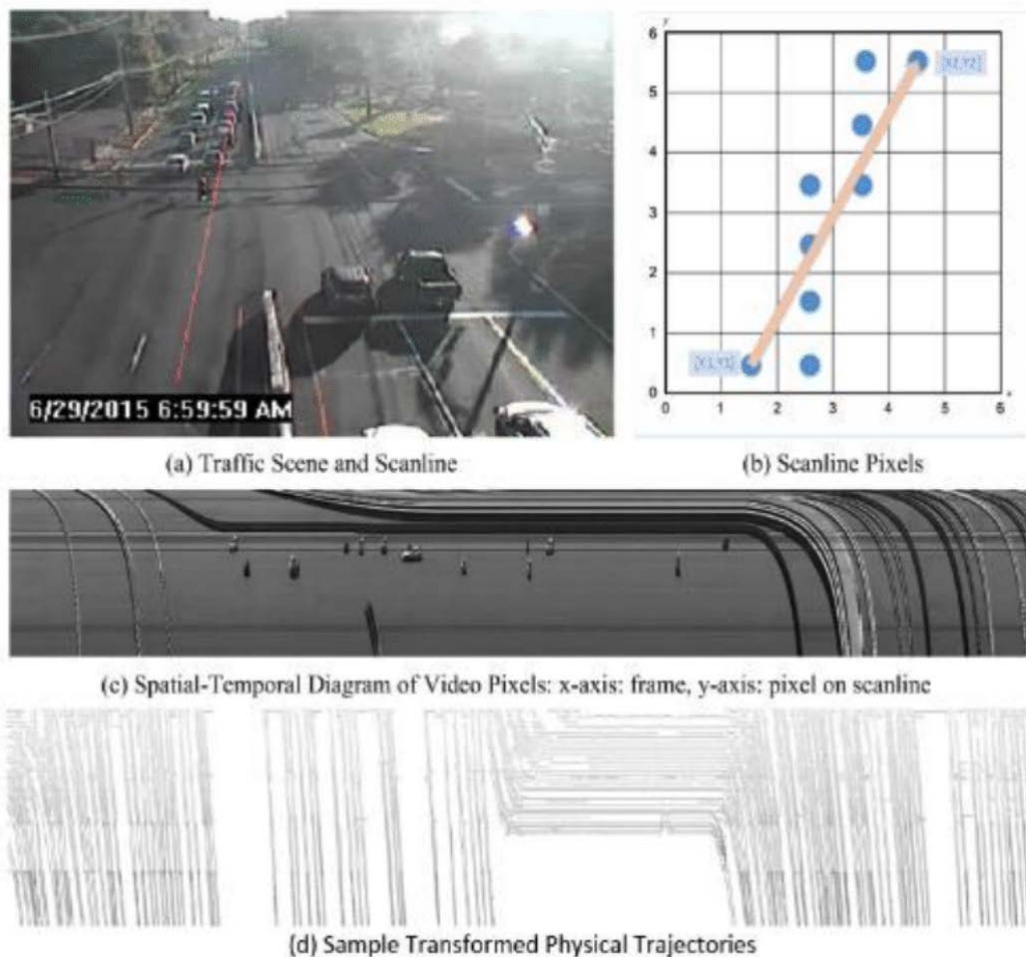


Figure 7 The Generation of STMap (Spatial-Temporal Map) From CCTV Video Input

Figure 7 describes the Spatial-Temporal map (STMap) generated from video input using a predefined scanline and leaving vehicle strands for each moving vehicle. The spatial-temporal map is a frame-by-frame

stacking of the pixel values on the scanline, showing the time progression of groups of pixels. Every moving object passing through the scanline will leave a group of strands. These groups of moving strands can be used to generate the groups of “pixel trajectories.” The following is the detailed definition of key STMap related features.

**Pixel:**  $(r_{p_i}, f_{p_i})$ . In a STMap, every pixel is RGB point and has the coordinates  $(r_{p_i}, f_{p_i})$ , where  $r$  means the  $r$ th row from top to bottom,  $f$  means the  $f$ th frame from left to right,  $p$  means the  $p$ -th strand,  $i$  means the  $i$ th point.

**STLine:** A Spatial-Temporal line (STLine) is generated from a scanline marked in a video that scans only one line rather than the whole view in order to reduce the consumption in computing resources. STLine is comprised of consecutive line segments marked in camera view that covers the detection range of the CCTV video. An appropriate STLine can preserve the trajectories of the vehicles passing through the corresponding lane, which enables it to use STMap for traffic monitoring. The intermediate pixels between STLine turning points are obtained through the DDA or Bresenham line drawing algorithm. To get the traffic state from CCTV camera video, STLines should be predefined in traveling lanes so that the whole trajectory of vehicles can be reflected by accumulating the pixel progressions on STLines. The vertical axis shows STLine from  $(x_{l1}, y_{l1})$  to  $(x_{lm_i}, y_{lm_i})$  and the horizontal axis shows different frames, one unit for one frame.


**STMap:** As Figure 7(c) shows, STMap is created by stacking pixel values on STLines for each frame from the video feeds. Time-Lapse pixel changes of STLines caused by moving vehicles will lead to distinctive strands on STMap. These groups of strands can be used to generate pixel trajectories to show the traces of vehicles to record vehicle movements on monitoring CCTV camera, which contains both the accurate time and distance measurements. The vertical axis of a STMap represents the traveled distance along predefined STLine and the horizontal axis of an STMap represents the frame number.

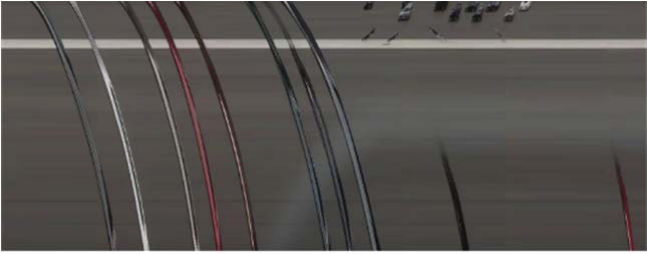
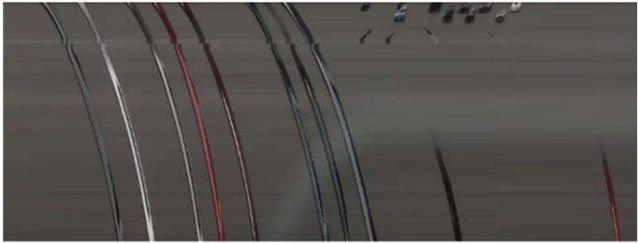
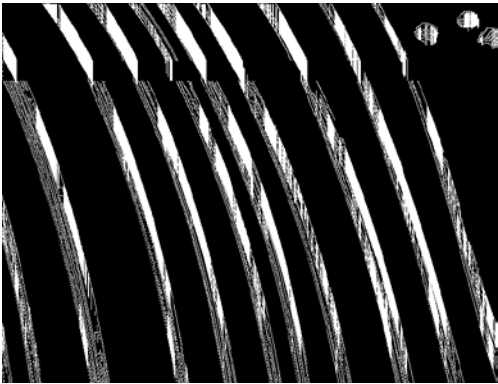

**Strand:** Every vehicle that passes through a STLine will leave a special pattern on the STMap, which is named as a strand, e.g. the black pattern in Figure 7(c). Strands are quite different from the background and can be regarded as evidence of vehicle passing. Every vertical line in a strand of STMap is left by a vehicle or more. With frame changing, the vehicle strands move from top to bottom and show clearly the vehicle movement. The color of strands also depends on the vehicles’ color.

#### 4.2. The Proposed STLine-based Video Analytic Algorithms

The proposed STLine-based video analytic algorithms include the following key processing steps. Some example processing results are provided to illustrate the outputs for each step.

Table 1 Key modules and sample outputs of the proposed video analytic algorithms

Processing Methods/Modules	Sample Output
a. Scanline based Camera and Lane Direction Determination	

<p>b. Spatial-Temporal Diagram</p>	 A spatial-temporal diagram showing the movement of vehicles over time. The vertical axis represents space (road lanes) and the horizontal axis represents time. Multiple colored lines (grey, red, blue) represent the trajectories of individual vehicles, showing their paths across the road lanes as they move through the scene.
<p>c. ST Diagram Denoising and Preprocessing</p>	 The same spatial-temporal diagram as in row b, but with significant noise removed. The trajectories are now much cleaner and more distinct, with the background being a uniform dark grey, making the colored lines stand out clearly.
<p>d. Vehicle Strands Detection from ST Diagram</p>	 A binary (black and white) image showing the detected vehicle strands. The strands are represented as thick, white, curved lines against a black background. The lines are parallel and follow the same general downward-sloping trajectory as seen in the previous diagrams.
<p>e. Crossing Vehicle Removal from ST Diagram</p>	 A binary image similar to row d, but with any crossing or overlapping strands removed. The remaining white strands are clean, parallel, and do not intersect, representing the final processed vehicle trajectories.

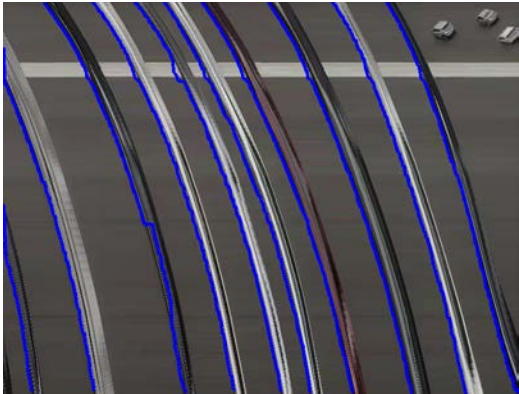

<p>f. Strand Edge (Front bumper location) Detection and Pixel Trajectory Output</p>	
<p>g. Occlusion Detection and Separation</p>	

Table 1 shows the major modules from the proposed model as well as the intermediate images from model outputs. In Table 1, Figure a is the matching results of a predefined scanline (in green) with the camera location. Figure b shows the generated STMaps from a time interval. Figure c shows the preprocessing outputs by identifying and removing horizontal noises from STMaps. Figure d connected edges using the heuristic analytics algorithm. Figure e is refined strands after removing crossing vehicles. Figure f shows the extract trajectories that represent detected vehicles from video input. Figure g shows the result of occlusion detection and separation, it's easy to see that the connected strands have been separated successfully based on a different color.

### 4.3.STLine Marking



Figure 8 Examples of STLines in Camera View

Before the counter can work, STLines have to be marked manually. A python script is used for STLine generation. It can read & check the RTMP links and capture & display the snapshots from the cameras one by one. Then the operator is able to mark the STLines on the snapshots by three steps: 1. Clicking left mouse to put inflection points of each STLine; 2. Pressing 'Space' to confirm the current STLine and start to mark the next one; 3. Pressing 'J' to save the STLines on the current snapshot and jump into the next snapshot. 'Space' and 'J' are chosen for they are friendly to both right hand operating and left hand operating. The STLines should be close to the most frequent trajectory of each lane, which requires experience-based human determination for now.

### 4.4.STLine-direction-based Camera and Lane Direction Determination

There are over 400 traffic cameras published to the New Jersey's 511 website by NJDOT and NJTA, each with its own installation respective to the roadways they are observing. These traffic cameras typically do not provide the direction they are facing which makes it difficult to determine the direction of traffic flow. The direction is needed in order to match the lanes in the video to the lanes in the field so that the counting results of two directions will not be reversed. One factor that should also be noted is that most of these traffic cameras have pan, tilt, and zoom capabilities and are being operated in real-time by different agency's Traffic Operations. This means the cameras can be panned, tilted, or zoomed at any given moment which can cause the direction of traffic with respect to the video feed to change.

In order to begin simplifying this problem, it was decided that NJDOT cameras would be focused on first. The NJDOT cameras were then grouped into 7 configurations by locating these cameras on Google Earth and determining which of the 7 configurations would fit best for each camera.

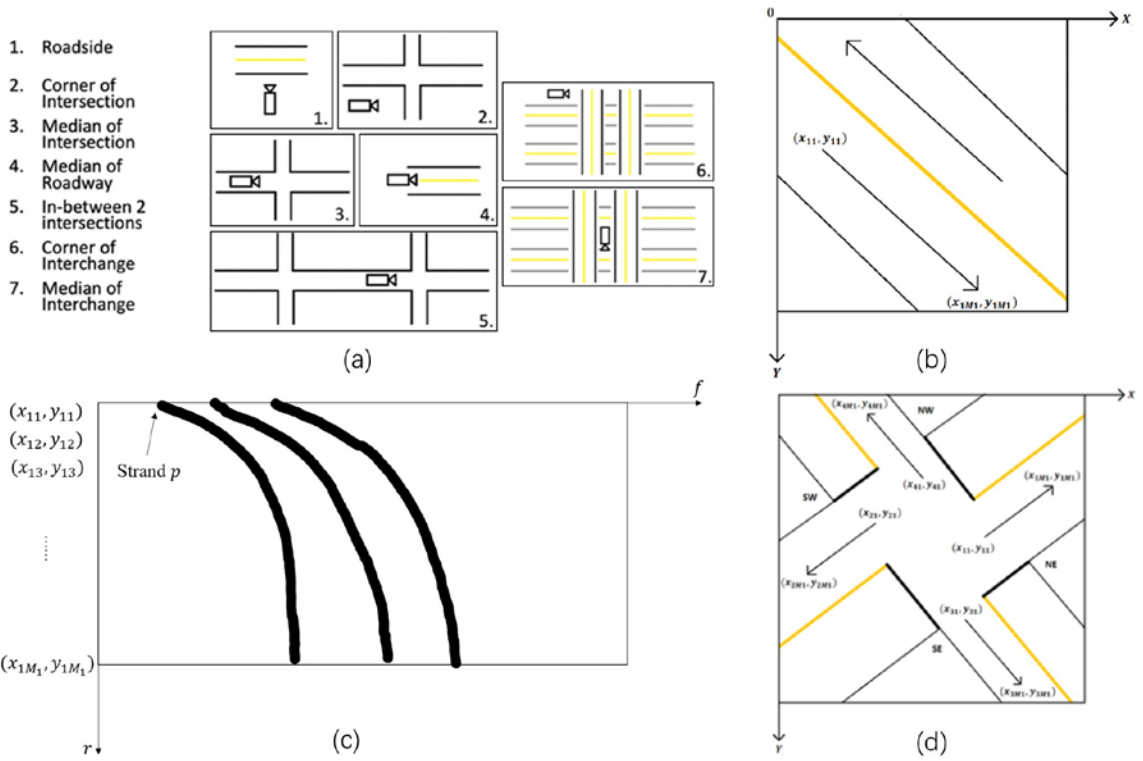


Figure 9 STLine-the connection between Camera View and STMap for Camera Direction Determination

Figure 9(a) above provides a visual representation as well as the naming convention for each of these 7 configurations.

Next, a reference table was created determining which of the seven configurations a camera fits into as well as the geographic location of the camera with respect to the roadway, intersection, or interchange.

The Lane Direction Determination method in this paper is simple and can be used for direction determination for the cameras which have asymmetric scenes, such as type 1, 2 and 6. Two inputs are required in this method. The first one is the relative position of the cameras to the road. A reference table containing a relative position was used. The relative position was expressed in EB, SB, WB, NB for roadside cameras and SE, SW, NE, NW for corner cameras. The other one is the STLine with the same direction of vehicle movement. For cloud-based vehicle counting in this paper, the STLines were marked manually to save the computing resources, which consisted of points that start from where vehicles came from and end where vehicles left. The STLines have the potential to be generated from trajectory rather than manual marking if ignoring the computing resources it will take.

For a roadside camera installed on the EB side of the road, 2 directions of lanes will be in the camera view. For example, the STLine in the Figure 9(b) has a start point  $(x_{11}, y_{11})$  and endpoint  $(x_{12}, y_{12})$ . If  $x_{12} > x_{11}$ , which means the endpoint is on the right of the start point, this STLine is on the same side with the camera, EB. If  $x_{12} < x_{11}$ , WB.

For a camera installed at the southeast (SE) corner of an intersection or interchange, 4 directions of lanes will be in the camera view. For example, the STLine in the Figure 9(d) has a start point  $(x_{11}, y_{11})$  and endpoint  $(x_{1M1}, y_{1M1})$ . If  $x_{1M1} > x_{11}, y_{1M1} > y_{11}$ , the STLine is on an EB lane. The whole relationship is in the table below.

Table 2 Relations between Camera Directions and STLine Pixel Coordinate Characteristics

Roadside Camera Position	Northbound Roadside		Southbound Roadside	
Start Point(x1,y1), Endpoint(x2,y2)	$x_{11} > x_{1M1}$	$x_{11} < x_{1M1}$	$x_{11} > x_{1M1}$	$x_{11} < x_{1M1}$
Direction of lane	SB	NB	NB	SB

Roadside Camera Position	Eastbound Roadside		Westbound Roadside	
Start Point(x1,y1), Endpoint(x2,y2)	$x_{l1} > x_{lM_l}$	$x_{l1} < x_{lM_l}$	$x_{l1} > x_{lM_l}$	$x_{l1} < x_{lM_l}$
Direction of lane	WB	EB	EB	WB
Corner Camera Position	Northeast Corner			
Start Point(x1,y1), Endpoint(x2,y2)	$x_{l1} > x_{lM_l}$ $y_{l1} > y_{lM_l}$	$x_{l1} > x_{lM_l}$ $y_{l1} < y_{lM_l}$	$x_{l1} < x_{lM_l}$ $y_{l1} > y_{lM_l}$	$x_{l1} < x_{lM_l}$ $y_{l1} < y_{lM_l}$
Direction of lane	SB	EB	WB	NB
Corner Camera Position	Southeast Corner			
Start Point(x1,y1), Endpoint(x2,y2)	$x_{l1} > x_{lM_l}$ $y_{l1} > y_{lM_l}$	$x_{l1} > x_{lM_l}$ $y_{l1} < y_{lM_l}$	$x_{l1} < x_{lM_l}$ $y_{l1} > y_{lM_l}$	$x_{l1} < x_{lM_l}$ $y_{l1} < y_{lM_l}$
Direction of lane	WB	SB	NB	EB
Corner Camera Position	Northwest Corner			
Start Point(x1,y1), Endpoint(x2,y2)	$x_{l1} > x_{lM_l}$ $y_{l1} > y_{lM_l}$	$x_{l1} > x_{lM_l}$ $y_{l1} < y_{lM_l}$	$x_{l1} < x_{lM_l}$ $y_{l1} > y_{lM_l}$	$x_{l1} < x_{lM_l}$ $y_{l1} < y_{lM_l}$
Direction of lane	EB	NB	SB	WB
Corner Camera Position	Southwest Corner			
Start Point(x1,y1), Endpoint(x2,y2)	$x_{l1} > x_{lM_l}$ $y_{l1} > y_{lM_l}$	$x_{l1} > x_{lM_l}$ $y_{l1} < y_{lM_l}$	$x_{l1} < x_{lM_l}$ $y_{l1} > y_{lM_l}$	$x_{l1} < x_{lM_l}$ $y_{l1} < y_{lM_l}$
Direction of lane	NB	WB	EB	SB



## 4.5. Static Noise Removal on STMap

Horizontal Processing Window

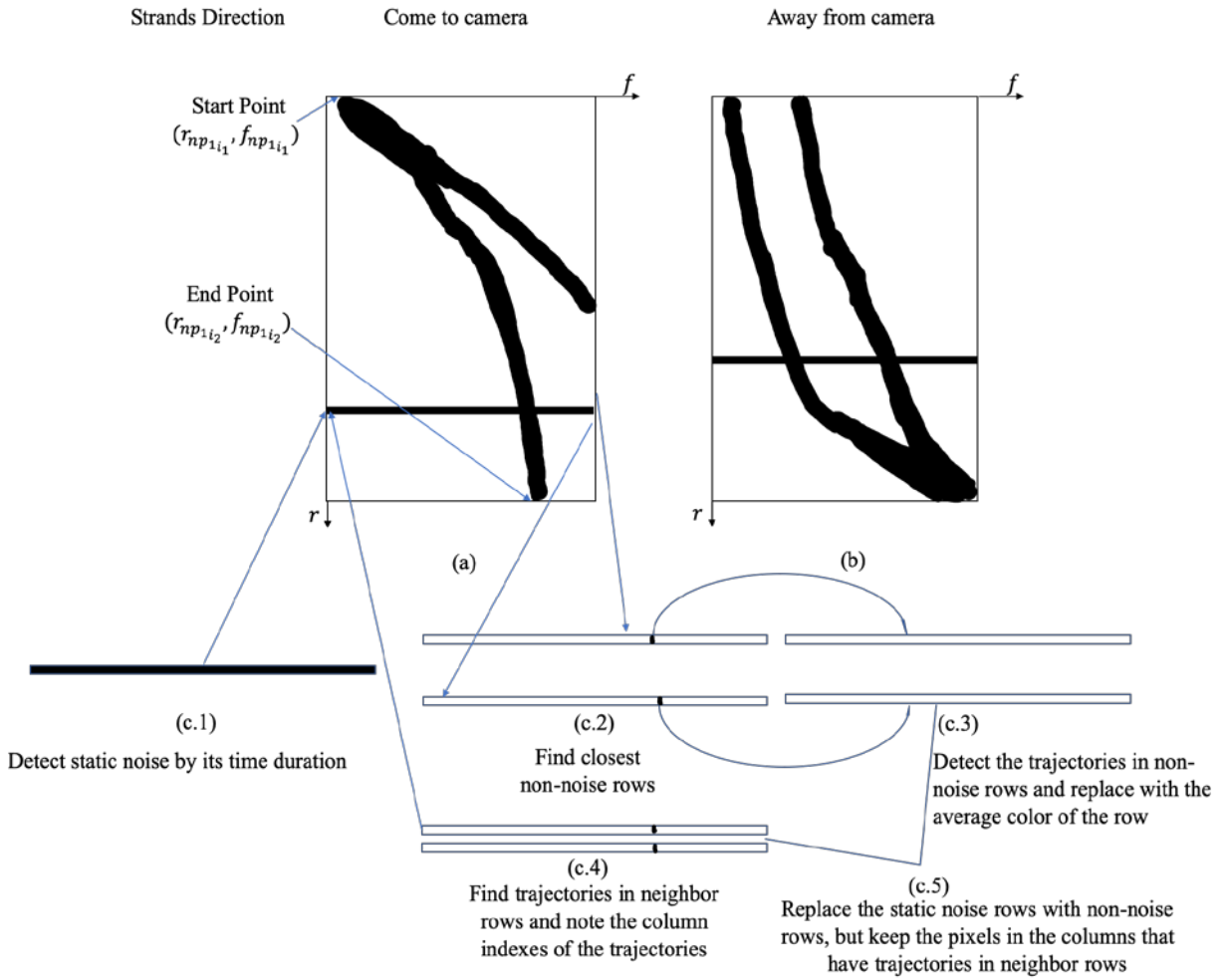


Figure 10 Illustration of the Static Noise (e.g. Light poles, lane markings, etc.) Removal Algorithms

As Figure 10(a)(b) shows, the static noise on STMap results from static objects usually stays longer in the same row, which has a significant difference with the strands caused by moving vehicles and can be detected by its duration and color difference. For example, in the camera view of US1 at NJ18, there is a pole covering all three westbound lanes and there are horizontal patterns in its STMaps of the three westbound lanes which do not have vertical movement at all and occupy the same rows for a long time.

**Detection:** For a processing window consists of  $T_{processing\ window}$  of video frames,

$$TD_p = \max(f_{p_i}) - \min(f_{p_i})$$

if  $TD_p > 0.8 \times T_{processing\ window}$  or ( $TD_p > 0.4 \times T_{processing\ window}$  and  $|RGB_p - RGB_{nr}| > thrld_{row\_diff}$ ), then  $p$  is static noise.

Where  $TD_p$  indicates the time duration of the strand  $p$ ,  $(r_{p_i}, f_{p_i})$  is the coordinate of the  $i$ th point in strand  $p$ ,  $r_{p_i}$  for vertical axis and  $f_{p_i}$  for horizontal axis,  $T_{processing\ window}$  indicates the total time of the processing circle,  $RGB_p$  indicates the mean RGB value of the strand,  $RGB_{nr}$  indicates the mean RGB value of neighbor row,  $thrld_{row\_diff}$  is a set value for RGB difference threshold.

**Removal:** To remove the static noise after the detection, a search for nearest non-noise-rows will be activated based on the detected static rows and the nearest non-noise-rows will be used to replace the static

noise rows after filling in the potential trajectory columns on the nearest rows without noises. Sobel vertical edge detector is used to detect the trajectory in nearest rows without noises and the detected trajectories will be replaced with the average color of nearest rows without noises from above and below the noisy rows. As Figure 10(c) shows, during the filling, if the noisy rows have vehicle trajectories on them, the blocks of pixels of vehicles will not be replaced to avoid removing critical trajectory information. If the filling rows, that is rows above or below the noisy rows without noises, have vehicle trajectories, the blocks of pixels of vehicles on those rows will be replaced by pixels before or after those blocks to avoid creating ghost vehicles.

---

For  $r_{noise}$  in  $R_{noise}$ ,

For row  $r$  from detected noise rows  $r_{noise}$  to both up and down boundaries,

if  $r \notin R_{noise}$ ,

use Sobel edge detector to extract the column indexes of vertical edges  $f_1, f_2, \dots, f_n$  in row  $r, r - 1$  and  $r - 2$ ,

$$\begin{aligned} C_{traj} &= \{r_{f_1}, r_{f_2}, \dots, r_{f_n}\} \\ r_{C_{traj}} &= AverageRGB(C_R C_{traj}) \\ r_{noise} &= r \end{aligned}$$

---

Where  $R_{noise}$  is the aggregation of the detected noise rows  $r_{noise}$ ,  $r_{f_1}, r_{f_2}, \dots, r_{f_n}$  are the columns of detected trajectory in row  $r$ ,  $C_{traj}$  is the aggregation of detected trajectory columns  $r_{f_1}, r_{f_2}, \dots, r_{f_n}$ ,  $r_{C_{traj}}$  are the corresponding columns of detected trajectory in row  $r$ ,  $R$  is the aggregation of all the columns on row  $r$ ,  $C_R C_{traj}$  is the complementary set of  $C_{traj}$ .

#### 4.6.STMap-based Vehicle Crossing Removal

Similar to Static Noise Removal, vehicle crossing can be detected by its time duration and height/width ratio too. The following is the proposed algorithm.

---


$$Size_p = (r_{p_i}) - \min(r_{p_i})$$

If  $TD_p < Thrld_{TD}$  and  $\frac{Size_p}{TD_p} > Thrld_{\frac{S}{T}}$ , or  $Size_p < Thrld_{size}$ ,

then strand  $p$  is vehicle crossing,

$$\begin{aligned} P &= \{p_1, p_2, \dots, p_i\} \\ P &= Color_{background} \end{aligned}$$

Where  $Size_p$  indicates the size of the object,  $Thrld_{TD}$ ,  $Thrld_{\frac{S}{T}}$  and  $Thrld_{size}$  are threshold values set manually,  $Color_{background}$  is the color of background.

---

#### 4.7.STMap-based Vehicle Occlusion Detection and Separation

From the roadside angle, vehicle occlusions can create significant issues for computer vision algorithms. In the previous HASDA model, due to the use of aerial video, the algorithm does not need to deal with severe vehicle occlusions which lead to significant undercounting of vehicles. In the proposed model, a new occlusion treatment method is proposed. The method is based on the observations that when vehicles are getting close to the camera locations, the separation among vehicles can be large enough for their strands to split on the STMap. The bisection method is used for occlusion separation. The detailed algorithm is as follows.

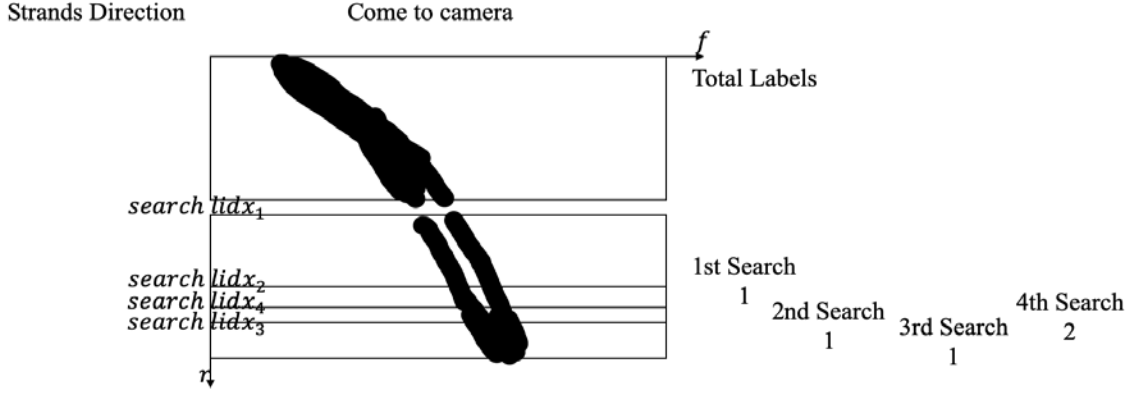


Figure 11 Illustration of the Proposed Occlusion Detection and Removal Algorithm

**Mean Frames based Occlusion Detection:** First, the black and white STMap  $bwm_{R_l \times F}$  (where  $R_l$  is the total number of pixels of the STLine on Lane  $l$ ,  $F$  is total number of video frames to be analyzed) is labeled by the connected components inside. To detect if there are any occlusions, mean frames that one label have are calculated:

$$mean\ frame = \sum_{r_{min}}^{r_{max}} \frac{len(fids_r)}{(fids_r) - min(fids_r)} \div (r_{max} - r_{min})$$

Where  $r_{max}$  is the maximum row index that label  $p$  has,  $r_{min}$  is the minimum row index that label  $p$  has,  $fids_r$  is the frame indexes that label  $p$  occupied on row  $r$ ,  $len(fids_r)$  is the total number of  $fids_r$ .

If  $mean\ frames < mean\ frames\ threshold$ , there exists at least one occlusion. Then the bisection-based occlusion separation is activated.

**Bisection-based Occlusion Separation:** The bisection-based occlusion separation will keep searching for splits within the search area chosen by bisection method. The first search area is the half STMap near the camera and it will keep being divided into 2 parts and the one close to camera will be searched until the splits occlude each other again or the search area is too narrow to supply valid information.

For search area  $n$  between  $search\ lid x_n$  and  $search\ lid x_{n-1}$  ( $search\ lid x_0$  is the boundary of camera side, e.g. the bottom of Figure 11)

Connect the components inside  $n$  and label the components, total labels number is  $p_n$ .

Group the  $p_n$ s together if the distance between any two of them is less than the separation threshold in case of over-separations,

Else,

Use Mean Frames based Occlusion Detection to check if there's occlusion inside the search area  $n$ ,

If there is occlusion inside  $n$ ,

$$search\ lid x_{n+1} = \frac{search\ lid x_n + reference\ lid x}{2},$$

Else,

$$reference\ lid x = search\ lid x_{n-1},$$

$$search\ lid x_{n+1} = \frac{search\ lid x_n + reference\ lid x}{2},$$

If  $p_n < P_n - 1$  or search area  $n$  is too narrow,

Stop and Return separated labels,

Else,

Search  $n + 1$ .

#### 4.8.STMap-based Vehicle Counting Combined with Lane-changing Detection

The proposed system effectively reduces the impact on vehicle counting caused by lane-changing. In the proposed system, the strands are only counted on the camera side for 3 reasons:

- The camera side has better resolution of the vehicle.
- The camera side has a better angle, at which the vehicles usually separate from each other.
- Only counting one side reduce the possibility of overcounting caused by lane-changing.

For strands that come to camera,

If  $\max(f_{np_i}) < 0.3 \times T_{processing\ window}$ ,

then the strand  $p$  belongs to a vehicle that merges out of current lane and it should not be counted.

Else,

$count += 1$ ,

Where  $\max(f_{np_i})$  is the horizontal index of the rightmost point in strand  $p$ ,  $\max(r_{np_i})$  is the vertical index of the bottommost point in strand  $p$ .

For strands that are away from the camera,

If  $(f_{np_i}) > 0.7 \times T_{processing\ window}$ ,

then the strand  $p$  belongs to a vehicle that merges in the current lane and it should not be counted.

Else,

$count += 1$ .

#### 5.Cloud computing architecture and system configuration

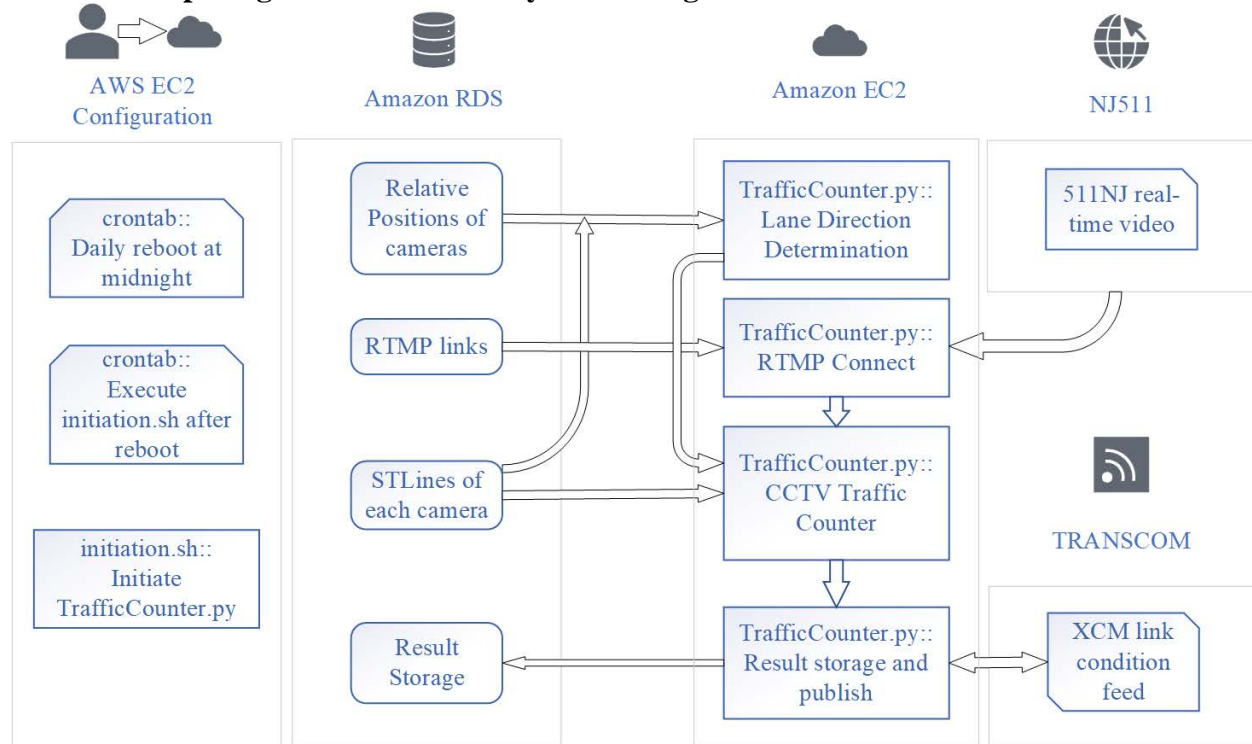


Figure 12 The Proposed Cloud Computing Architecture and System Configuration

In the test platform, the cloud computing services are implemented in the Amazon Web Service (AWS) platform. AWS is the industry-leading cloud computing services with its comprehensive toolsets, low pricing, user-friendly dashboards for managing different services, and the availability of documentation and examples. EC2 instances were used to analyze the real-time traffic video, extract useful information

from the video and transfer the information to the Oracle database deployed on Amazon RDS. Another EC2 instance was used to collect the database data and TRANSCOM data feed and publish them.

The chosen t2.micro instance with unlimited burst activated was launched which had a 2.5GHz vCPUs with 1GiB memory. Linux was installed because of its open-source, multi-functional, efficiency and stable characteristics. The operating system version installed on the AWS EC2 instances was Amazon Linux AMI release 2018.03, on which python 3.6.5 was installed. To ensure the stability for running over the years and save the cost, a daily auto-rebooting was set at 9 p.m., after which the light condition is not good enough for traffic detection. The video processing script would be initiated the next morning when the light condition is good.

### 5.1.Daily Reboot

The daily reboot is based on the CRON schedule. The default time zone is UTC-0, which is inconvenient and needs to be changed to Eastern Time Zone.

Firstly, `sudo vim /etc/sysconfig/clock` to edit the clock configuration file. Locate the *ZONE* entry and change it to *ZONE="America/New\_York"*, press *esc*, type in `:wq`, press *enter* to save the profile.

After changing the time zone, use `sudo crontab -e` to edit the system crontab file. Then, type in `0 21 * * * /sbin/reboot -h now`. Last step, press *esc*, type in `:wq`, press *enter* to save the profile. The detailed table of the 5 values set in crontab is below.

Table 3 Crontab Scheduling Parameters Configured in Amazon Web Services(AWS)

Field Name	Value Sample	Value Range	Description
Minute	0	0-59	The exact minute that the task will start.
Hour	21	0-23, *	The exact hour that the task will start. If use *, the task will start every hour.
Day of Month	*	1-31, *	The exact day(of the month) that the task will start. If use * for Day of Month and Day of Week, the task will start every day.
Month	*	1-12, *	The exact month that the task will start. If use *, the task will start every month.
Day of Week	*	0-7, *	The exact day(of the week) that the task will start. If use * for Day of Month and Day of Week, the task will start every day.

### 5.2.Daily Runner

Similar to Daily Reboot, Daily Runner is based on CRON schedule too. However, it uses `crontab -e` rather than `sudo crontab -e`. They are different files from two different directories.

Firstly, a bash file named `run.sh` has to be created in the directory `/home/ec2-user/`. The contents of the bash file should be in a format like this: `nohup python TrafficCounter.py 'I-80 at I-287' & . nohup` means no hang-up, which will keep the code running consistently. *Python* is the executor that will execute the script. *TrafficCounter.py* is the script. *'I-80 at I-287'* is the camera location that will be monitored. `&` is to let the task run in the background.

Secondly, the bash file has to be executable. Type in `chmod 777 run.sh` and punch *Enter*.

Thirdly, use `crontab -e` to edit the user crontab file. Type in `30 6 * * * . $HOME/.bash_profile; sh run.sh 2>>/home/ec2-user/log`. Then `esc+:wq+enter` to save the profile.

### 5.3. Python Configuration

Many python modules were required. The modules used to process the videos are as follows.

Table 4 Python Modules Deployed in AWS Cloud Computing Platform

Module name	Version	Function
numpy	1.14.3	The fundamental package for scientific computing with Python.
cv2	3.4.3	An open-source computer vision and machine learning software library.
csv	1.0	Read and write csv files.
scipy.ndimage.gaussian_filter	1.1.0	Gaussian filter.
cx_Oracle	6.4.1	Connect database.
time, datetime	N/A	Current time.
threading	N/A	Multi-threading.

The video analytic modules deployed in AWS completes the following tasks.

- **STLine Loading and Direction Determination:** The AWS EC2 instance reads the camera relative position table and STLine table from database to match all the STLines with the lanes based on the direction of STLines and generates the metafile for each camera containing information such as STLine coordinates, Lane directions, RTMP link, camera location. The AWS EC2 instance then uses the metafile to generate the STLines for all the lanes.
- **STMap Generation:** The EC2 instance gets the Real-Time Messaging Protocol (RTMP) link of the camera and reads the corresponding video from NJ511. Once the video is read successfully, it will use STLines to extract the pixels from the video and keep generating the STMaps.
- **STMap Processing:** The EC2 instance denoises and processes the STMaps, then the EC2 instance extracts the strands from STMaps.
- **Result Archiving:** The EC2 instance stores the result back to RDS every minute.
- **Flow Data Exporting and Publishing:** The EC2 instance reads the corresponding TRANSCOM data feed, adds the result to the feed and publishes it.

The first task is a pre-processed task and only needs to be run once, but the remaining four tasks have to keep running.

### 6. TransFusion Link Data Processing

TRANSCOM provides real-time travel time data feed over the road networks composed of road links. In general, TRANSCOM receives data from 1) TRANSCOM member agencies and centers in New Jersey, New York, and Connecticut; 2) Travel time data from E-ZPass electronic toll; 3) Third-party transportation Data Providers including from INRIX and HERE(Nokia). TRANSCOM uses its Data Fusion Engine to validate and aggregate real-time and historical information (e.g., speed and travel time) referenced to a local transportation network. In addition, TRANSCOM converts these inputs to a geographically universal format.

Table 5 is the list of selected routes in Northern NJ, which are manually processed to acquire the upstream and downstream order among road links. TransFusion Link Data has 65,667 links in New Jersey road network, and 5,855 of them (major parts of freeways in north New Jersey) were selected.

Table 5 Selected Links used for the Evaluation of the Proposed Platform

Route	Selected Part	Route	Selected Part
NJ 3	All	I-95	All
NJ 4	East of Paramus Road	I-195	All
NJ 17	North of 3 to 287	NJ 208	All
NJ 18	138 to Paulus Blvd & Paulus Blvd to River Road	I-278	All
NJ 21	North of 280	I-287	All
NJ 24	All	NJ 440	Bayone bridge to 78 & East of 95 to outerbridge crossing
I-78	All	NJ 495	All
I-80	All		

### 6.1.ArcGIS-based Direction Processing

To generate the direction of TransFusion Links accurately based on the New Jersey’s Standard Route Identifier (SRI) system, the following steps need to be completed in ArcGIS:

#### 6.1.1.Feature to line Tool

Since there are small gaps between adjacent links generated based on TransFusion link data, links need to be connected for manually creating routes. *Feature To Line* tool is used to connect Transcom links in preparation for the next step.

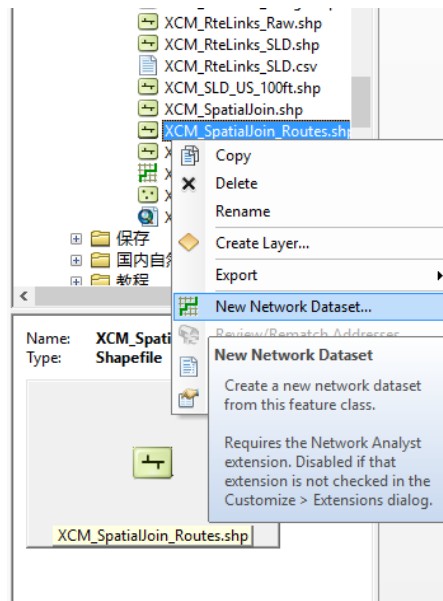


Figure 13 Create New Network Dataset for Matching with the TRANSCOM Link System

#### 6.1.2.Create New Network Dataset

*Network Analyst Tool* is selected to create routes. Right-click the shapefile generated in Step 1, then create a new network dataset.

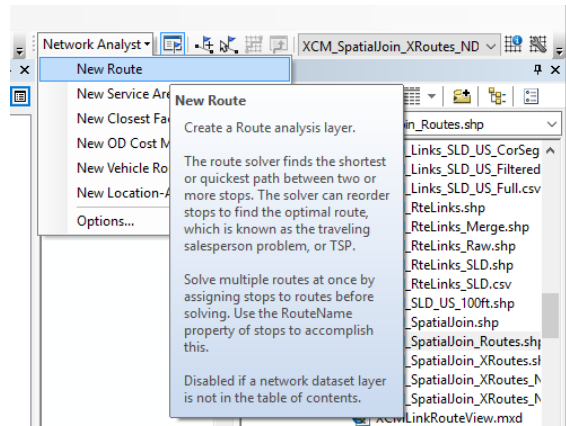


Figure 14 Create New Route for Geospatial Matching with CCTV Traffic Camera Locations

### 6.1.3. Create New Route

Go to Customize => Toolbar => Network Analyst. A *Network Analyst* Toolbar will pop up. Then create a new route layer by using this toolbar, as shown in the figure below.

Create stops for the desired route using *Create Network Location Tool* and create the desired route passing these stops using *Solve*.

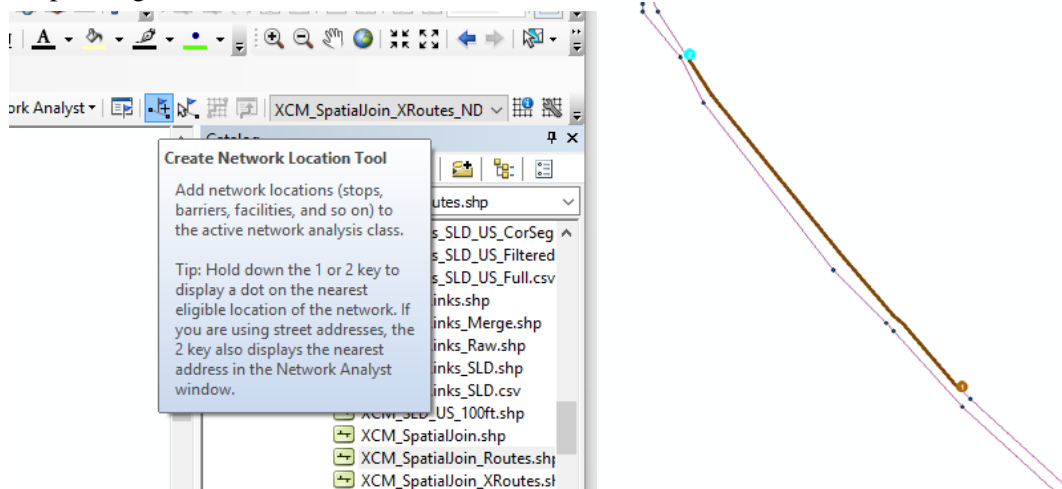


Figure 15 Export Directions of Routes for Geospatial Matching with CCTV Traffic Camera Locations

### 6.1.4. Export Directions of Routes

Use the *Directions* tool in *Network Analyst* Toolbar. If there is a route created in the network, the direction window will pop up, containing information about which links have been passed, as shown in the figure below. The direction of the route can be saved as text format, showing the LINKID of links along the corresponding route.



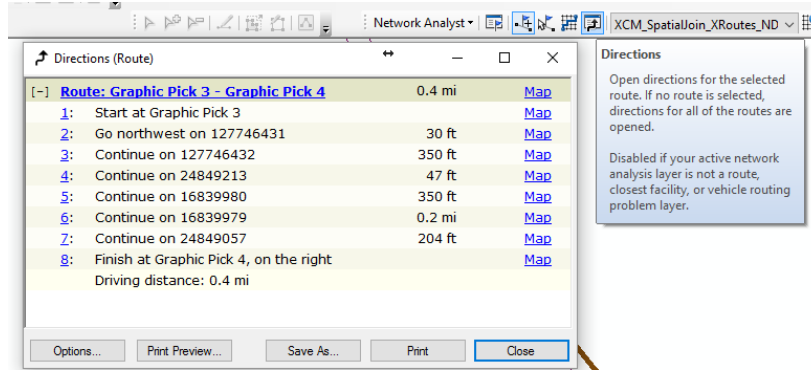


Figure 16 Final Direction Results Generated for Geospatial Matching with Camera Locations

### 6.2. Matching between TransFusion Links and CCTV Cameras

The road links are matched with CCTV cameras based on the camera locations. Road links that are matched with cameras if they are main roads and within 200 feet of the cameras. The following figure shows an example at the intersection of I-78 and NJ-24. The camera is “I-78 at NJ 24” facing towards I-78, therefore, surrounding road links of I-78 are matched with this camera. What’s more, the TransFusion links also have the direction information, which is also stored and will be used for the direction matching in the STMap.



Figure 17 Matching Example between the Cameras and its Adjacent Links

## 7. Traffic Database Management System

### 7.1. Database Management

The database was deployed on Amazon RDS db.t3.micro instance, which had 2 vCPU and 1GB RAM. The version of the database was Oracle 11.2.0.4.v17, which could work with the python module cx\_Oracle.

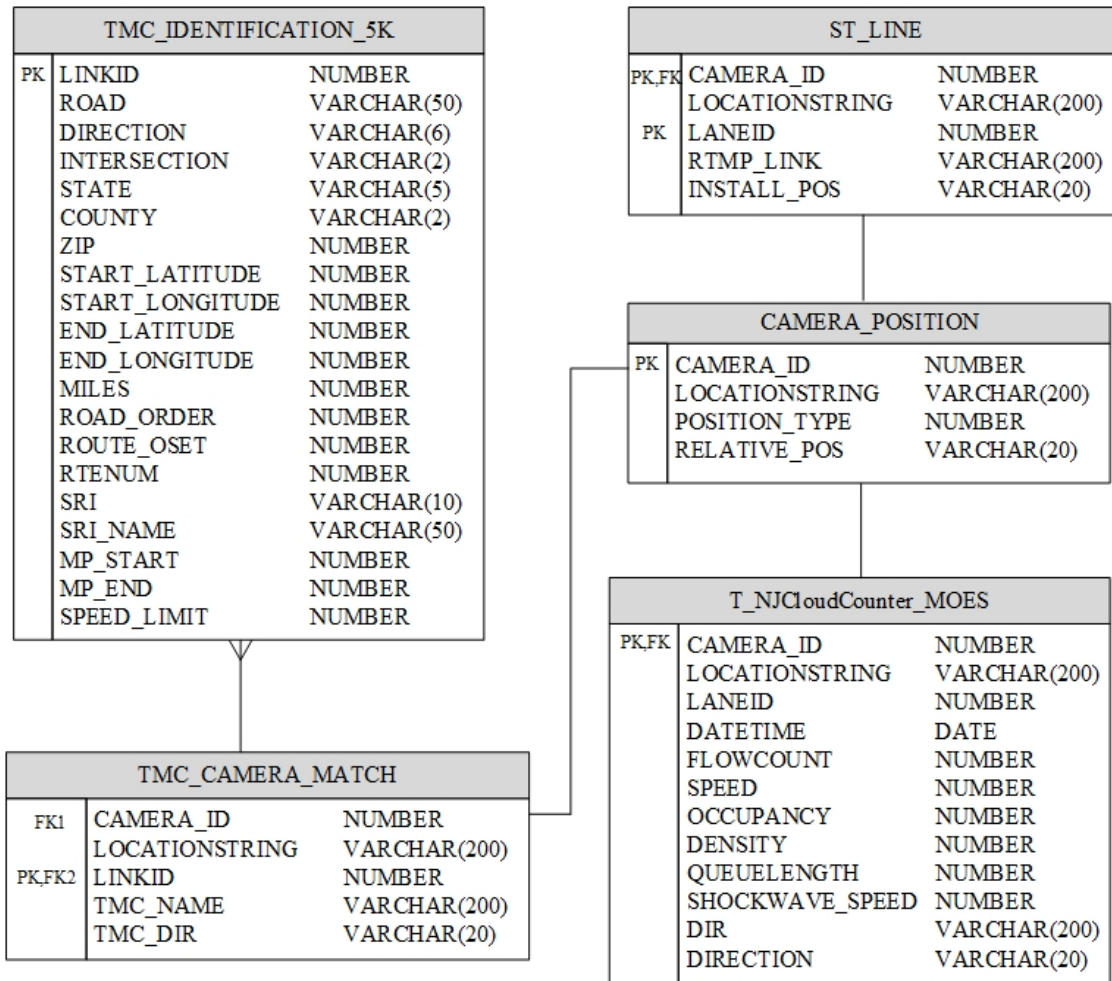


Figure 18 Database Entity-Relationship Diagram (PK: Primary Key; FK: Foreign Key) for Database Tables used in the Proposed Platform

Figure 18 is the Entity-Relationship Diagram of the database. The database was used for the storage of pre-marked STLines, relative position table, and the generated results. The relative position table contains the installation information of the cameras, which could be used for lane direction determination. The main table is the CAMERA\_POSITION table with information on their names and installs positions. TMC\_CAMERA\_MATCH is related to two tables, including 1) one-to-one relate to CAMERA\_POSITION table on its CAMERA\_ID field, and 2) one-to-many relate to TMC\_IDENTIFICATION\_5K on the LINKID field. Table ST\_LINE and Table T\_NJCloudCounter\_MOES relate to the CAMERA\_POSITION on the CAMERA\_ID field. The following is the Table field information and their corresponding description.

Table 6 Data Fields in TMC\_IDENTIFICATION\_5K Table  
Description: Selected TMC links in North NJ.

Field Name	Field Type	Field Name	Field Type
LINKID	NUMBER	END_LONGITUDE	NUMBER
ROAD	VARCHAR2(50 BYTE)	MILES	NUMBER
DIRECTION	VARCHAR2(6 BYTE)	ROAD_ORDER	NUMBER
INTERSECTION	VARCHAR2(2 BYTE)	ROUTE_OSET	NUMBER

STATE	VARCHAR2(5 BYTE)	RTENUM	NUMBER
COUNTY	VARCHAR2(2 BYTE)	SRI	VARCHAR2(10 BYTE)
ZIP	NUMBER	SRI_NAME	VARCHAR2(50 BYTE)
START_LATITUDE	NUMBER	MP_START	NUMBER
START_LONGITUDE	NUMBER	MP_END	NUMBER
END_LATITUDE	NUMBER	SPEED_LIMIT	NUMBER

Table 7 Data Fields in CAMERA\_POSITION Table  
Description: Relative position table sample

Field Name	Field Type
Camera_ID	NUMBER
LOCATIONSTRING	VARCHAR(200)
POSITION_TYPE	NUMBER
RELATIVE_POS	VARCHAR(20)

Table 8 Data Fields in TMC\_CAMERA\_MATCH Table  
Description: the TransFusion Links matched with cameras

Field Name	Field Type
Camera_ID	NUMBER
LOCATIONSTRING	VARCHAR(200)
LINKID	NUMBER
TMC_NAME	VARCHAR(200)
TMC_DIR	VARCHAR(20)

Table 9 Data Fields in ST\_LINE Table

Description: The pre-marked STLines metadata for each camera included STLine information such as Lane ID, Lane Direction(Calculated using model 4.2 with relative position table), STLine points and Real-Time Messaging Protocol (RTMP) link, based on which the script read the real-time video stream and generated the STMaps, one STMap per lane. Table 9 shows a sample of the metadata.

Field Name	Field Type
Camera_ID	NUMBER
LOCATIONSTRING	VARCHAR(200)
LANEID	NUMBER
RTMP_LINK	VARCHAR(200)
INSTALL_POS	VARCHAR(20)

Table 10 STLines Data Structures

Lane ID	Lane Direction	$x_1$	$y_1$	...	...	$x_M$	$y_M$
Lane 1	SB	$x_{11}$	$y_{11}$	...	...	$x_{1M}$	$y_{1M}$
...	...	...	...	...	...	...	...
Lane $l$	NB	$x_{l1}$	$y_{l1}$	...	...	$x_{lM}$	$y_{lM}$
RTMP link	rtmp://****.***.***/live/xxxxx_nj						

Installation Position	WB
Camera ID	7
Camera Location	XX @ YY

Table 11 Data Fields in T\_NJCloudCounter\_MOES Table

Description: The processing script could generate the STMaps based on the pre-marked STLines and detect the vehicles, with which the public TRANSCOM feed could be added and republished. The Location String, Land ID, Direction, Datetime, Flow Count, Speed, Occupancy, Density, Queue Length, Shockwave Speed extracted from videos and read from TRANSCOM feed were stored in Oracle database.

FIELD_NAME	FIELD_TYPE
LOCATIONSTRING	VARCHAR2(200 BYTE)
LANEID	NUMBER
DATETIME	DATE
FLOWCOUNT	NUMBER
SPEED	NUMBER
OCCUPANCY	NUMBER
DENSITY	NUMBER
QUEUELENGTH	NUMBER
SHOCKWAVE_SPEED	NUMBER
DIR	VARCHAR2(200 BYTE)
DIRECTION	VARCHAR2(20 BYTE)

## 8. System Calibration and Evaluation

### 8.1. Model Evaluation

$$E = \frac{1}{M_{Total}} \sum_1^{M_{Total}} (C_{GT} - C_D)$$

$$MAE = \frac{1}{M_{Total}} \sum_1^{M_{Total}} |C_{GT} - C_D|$$

$$MAPE = \frac{1}{M_{Total}} \sum_1^{M_{Total}} \frac{|C_{GT} - C_D|}{C_{GT}}$$

Where  $M_{Total}$  is the total minutes,  $C_{GT}$  is the ground truth count,  $C_D$  is the detected count.

Mean Error is calculated to evaluate the undercounts and overcounts of the proposed algorithm through the whole video. The more it is close to 0, the better the result is. Mean Absolute Error and Mean Absolute Percentage Error is used to evaluate the average performance and stability of the proposed algorithm in every minute. The lower the better. Ground truth vehicle counts are manually counting results that have been grouped by a time interval to reduce the impact of human reaction time.

### 8.2. Key Parameters

For vertical threshold which stands for length, ratios are used because the length of STLines may vary with the cameras' height, angle, and resolution. For the horizontal threshold which stands for time, values are used because the frame rate is consistently 8 frames per second. The noise ratio threshold is the threshold

parameter to detect static noises. For each row of a STMap, if the occupancy of this row after horizontal edge detection and dilation operation is greater than this threshold, then it will be considered a row with static noise. Time frame duration is the parameter to filter small blocks. Patterns that are either longer than this parameter or wider than this parameter will not be filtered. The time difference threshold is the parameter for the time difference detection algorithm. Based on the assumption that the intensity of moving object changes faster than the background, this parameter is used to segment out vehicle strands on STMaps. The threshold parameter to avoid overcounting caused by lane changes. The lane changes happen before the threshold will be removed but the incomplete strands that end before the threshold because of other issues will also be removed.

Table 12 Key parameters

<b>Parameter List</b>	<b>Descriptions</b>
Noise ratio threshold	The recommended range is 0.2-0.6 for all lanes.
Time frame duration	The preset value is 15 and the recommended range is 10-20.
Time difference threshold	The preset value is 15 and the recommended range is 10-20.
Lane change threshold	The recommended range for this threshold is between 0.3 and 0.7.

## 9. System Evaluation and Pilot Testing Results

### 9.1. CCTV Traffic Camera Data Sources

New Jersey's 511NJ system is an Advanced Traveler Information System that is available by phone or web 24 hours a day, 365 days a year. The information posted to this system is gathered by multiple public agencies including the New Jersey Department of Transportation and consists of information including but not limited to travel times, incident and construction information, as well as live traffic video. The 511NJ traffic video streams provide over 450 real-time traffic feeds to the motoring public and include video streams from NJDOT as well as the New Jersey Turnpike Authority. These video streams are generated from permanent traffic cameras installed by both organizations to help monitor traffic conditions and monitor New Jersey's roadways for incidents. These cameras installed along interstates, highways, and arterials are located at key strategic locations determined by Traffic Operations from each agency.

NJ18 at US1

US1 at Henderson Rd.



Figure 19 Camera View and STLines

The recorded videos used in this paper are from the 511NJ system which has a resolution of 320\*240. The 12-min-long video tested in this paper was captured around 12:05 P.M. on 21<sup>st</sup> July 2019 from a roadside camera of NJ18 at US1. It was sunny with good lighting. The other video tested in this paper was captured around 10:40 A.M. 10<sup>th</sup> April 2018 from an intersection camera of US1 at Henderson Road.



Figure 20 Snapshot of the VLC Traffic Counter for Generating Ground Truth Data

The ground truth data used was obtained through manual counting on traffic videos. The raw ground truth consisted of the time and the lane number of each vehicle passing and was grouped into 1-min-interval or 5-min-interval traffic counts according to demand.

The native test device is a 15-inch MacBook Pro which has an Intel 8850H@2.6GHz CPU with 32G RAM and the system is macOS Mojave 10.14.5. The cloud test server is AWS EC2 t2.micro instance, which has a 2.5GHz vCPUs with 1GiB memory and the system is Amazon Linux AMI release 2018.03. The evaluation was based on 1 min count.

**9.2.Video Analytic Model Validation Results**  
**9.2.1.ST-Line Creation and ST Map Generation and Denoising**



(a)NJ18 at US1 with STLines



(b)Raw STMap



(c)Static Noise Free STMap

Figure 21 STMap Generation from Traffic Video with Pre-marked STLines

The STLines marked in Figure 21(a) were used to generate the STMap. Figure 21(b) is the STMap of lane 4 between the 4801st frame and 5280th frame from a 12-min-long video recorded from NJ18 at US1. The horizontal axis of a STMap consists of frames and the vertical axis consists of STLine points. The raw STMap cannot be directly used for vehicle detection before some appropriate denoise processing. It is easy to find that the static noise in Figure 21(b) was caused by the black pole in Figure 21(a). The system does not process the whole camera view but detects static noise in STMap by its color difference and time duration. Figure 21(c) shows the Static-Noise-Free STMap, in which the static noise rows in Figure 21(b) have been detected and filled with the searched clean rows. Most of the trajectory columns in the filled row have been detected and replaced with row's average color. By comparing Figure 21(a) and Figure 21(b), it is obvious that the static noise was cleaned effectively and some of the missing part of strands was fixed, which allowed the following steps to extract the strands and count vehicles. There are some pixels kept in static noise rows because they have trajectory neighbors.

### 9.2.2.Strand Clustering Processing Results

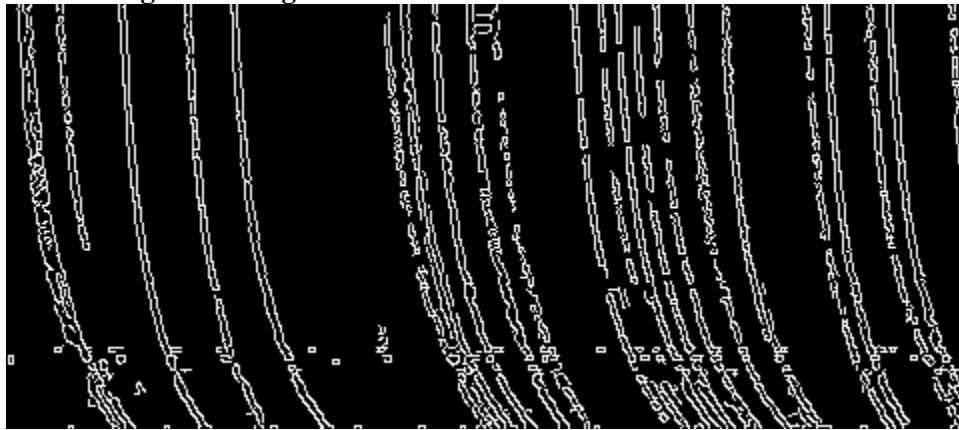


Figure 22 Sample output of Canny Edge Processing Algorithms for the STMap

Canny edge detector is used for edge detection to extract strand from denoised STMap (Figure 21(c)). From Figure 22, we can easily find that there are unregarious edges around the detected static noise rows. Those are caused by the pixels kept with potential trajectories determined by their neighbor rows, among which 51/63 are in real strands. Generally speaking, canny edge detector decently completed its job and filtered the irrelative background, but further steps are required to separate the connections between strands and extract the strands correctly.

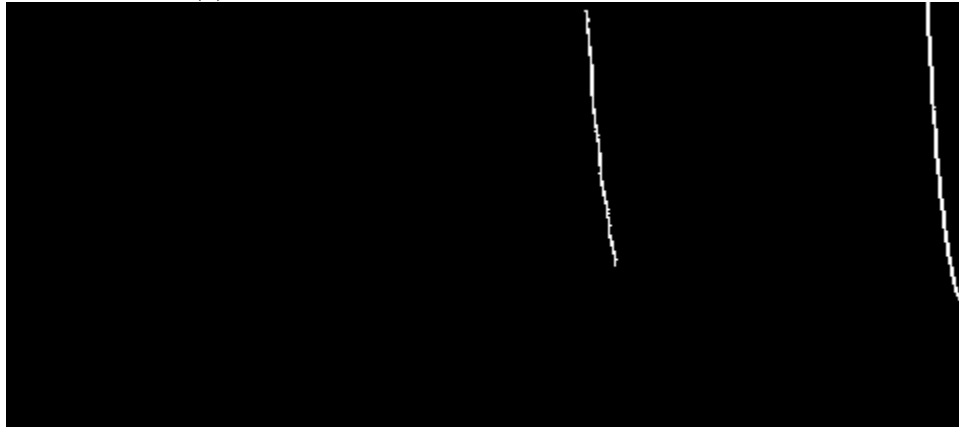


(a)Time difference: Time difference threshold=10





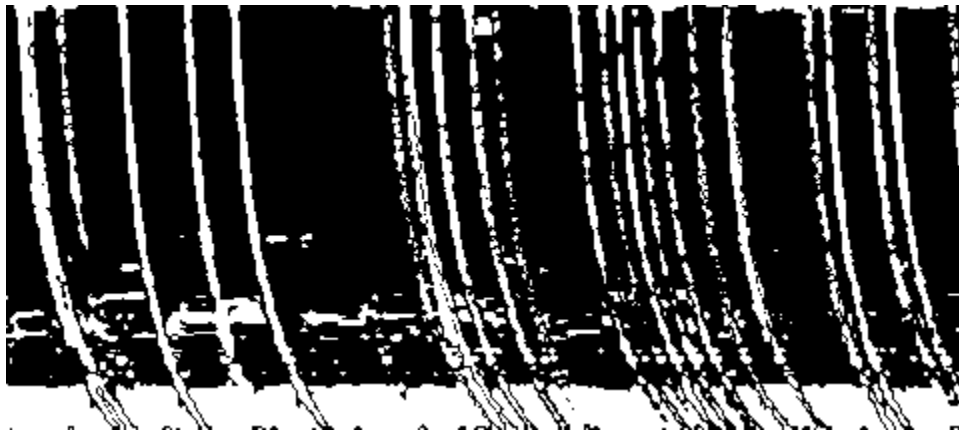
(b)Time difference: Time difference threshold=15



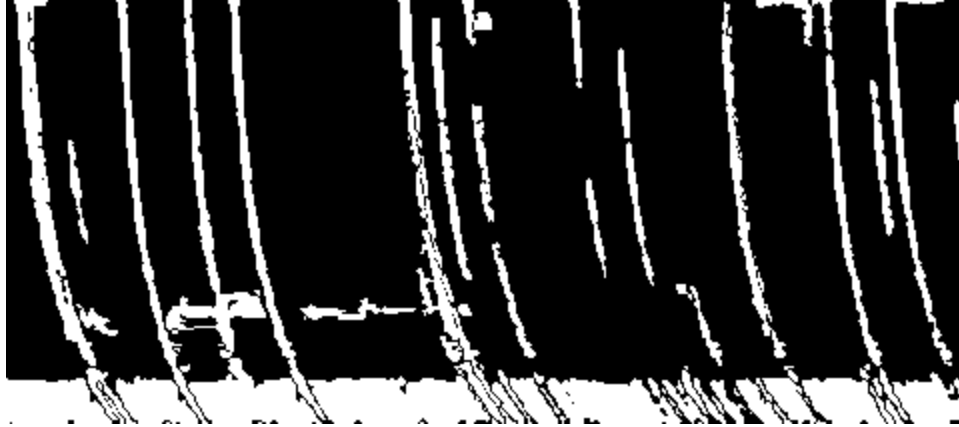
(c)Time difference: Time difference threshold=20

Figure 23 Sample Output of Time Differencing Processing of the STMap

The time-difference-based motion detector is used as a reference for strand extraction. The time difference algorithm is based on the assumption that the change of background is slower than the change between background and the vehicle strands within a time interval. From Figure 23(a)(b)(c), 10 is too small for a threshold which fails to filter all the noise. Although the one with 15 as threshold only detects 6 strands, the detected strands are all correct, which is acceptable for a supplement. 20 is too big that keeps only 2 strands.



(a) Sample Output of Thresholding



(b) Sample Output of Thresholding After Filtering the Smaller Components  
Figure 24 Sample Output of Thresholding

As another source of reference, thresholding is used to separate the roadway background and vehicle strands based on the assumption that they occupy different ranges of the intensity value. It used the denoised STMap as input and use thresholding method to separate different parts in the STMap. The threshold values are generated automatically using the triangle method, which resulted in the misdetection at the bottom of the Figure 24(a). The removal of static noise cuts the background apart, which made the triangle method separated two parts of background too. As Figure 24 shows, there are a lot of small noise detected because their intensity values exceed the threshold value. Therefore, a size filter was applied to filter the small noise, as Figure 24(b) shows.

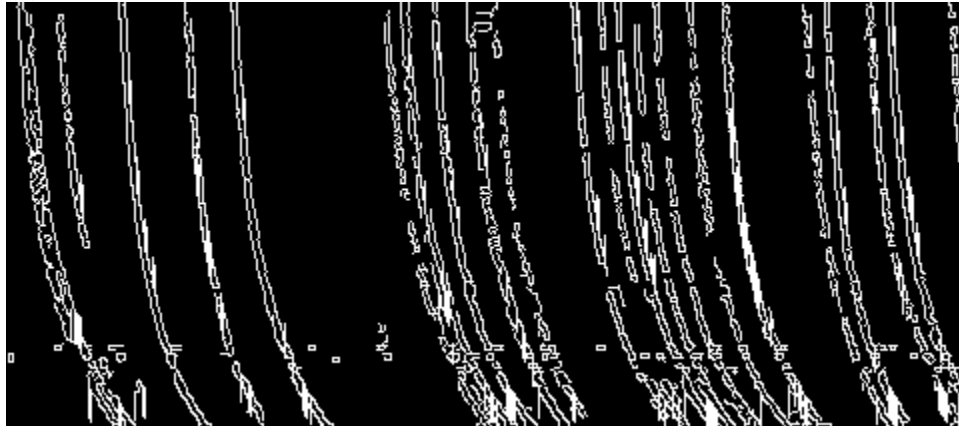


Figure 25 Sample Output of Canny Combined with Time Difference and Threshold  
The output of time difference module and threshold module were then combined together as a mask to process the output of canny edge detection, as Figure 25 shows.

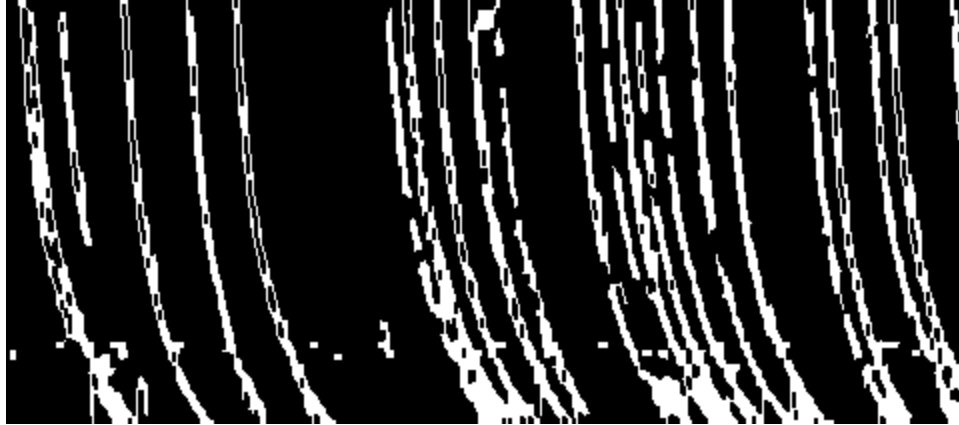


Figure 26 Sample Output of Filled Canny

The edges are not enough for strand extraction. Before strand extraction, the edges have to be filled, as Figure 26 shows. After filling, there's still small noise around the static noise rows which should be removed and can be removed based on their duration.

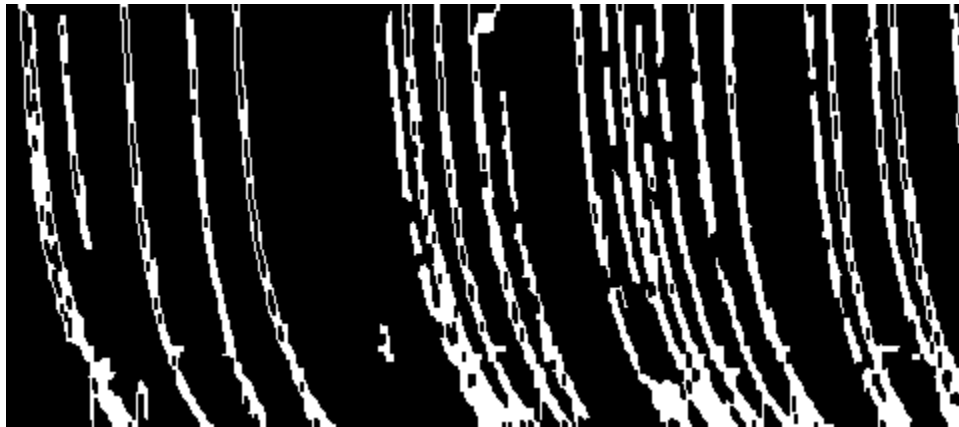


Figure 27 Sample Output after Removing Noise by Duration

A duration-based filter is used to remove the small noise in Figure 27 left by the conservative static noise removal.

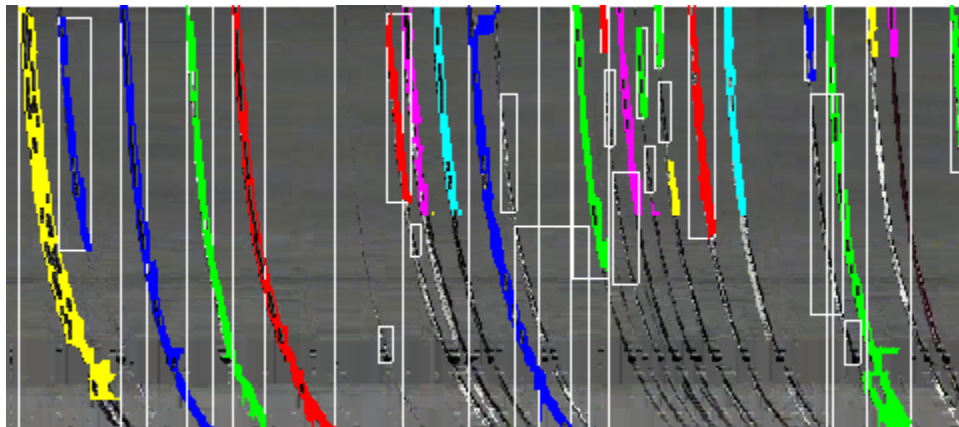


Figure 28 Sample Output of Counted Strands

Figure 28 shows the result of strand detection. Most strands including the smaller ones caused by lane-changing have been detected successfully. However, there is still over-counting caused by inconsistent strands. Combining with the raw STMap, the separation results from the strong change in color, which may be the effect of irregular vehicle movement. Generally, the strand extraction and vehicle detection have decent performances.

### 9.3.Lane Direction Results

Using the Table 2 combined with camera relative position table, the directions of the lanes in 220 of 444 NJDOT cameras of type 1, 2, 6 can be determined automatically and the directions of the lanes in other 11 cameras of type 3, 5 can be determined easily by checking whether the camera is facing the intersection or not. The rest cameras may require other solutions to match lanes manually such as comparing the features in the camera view with the features in Google Street Map.

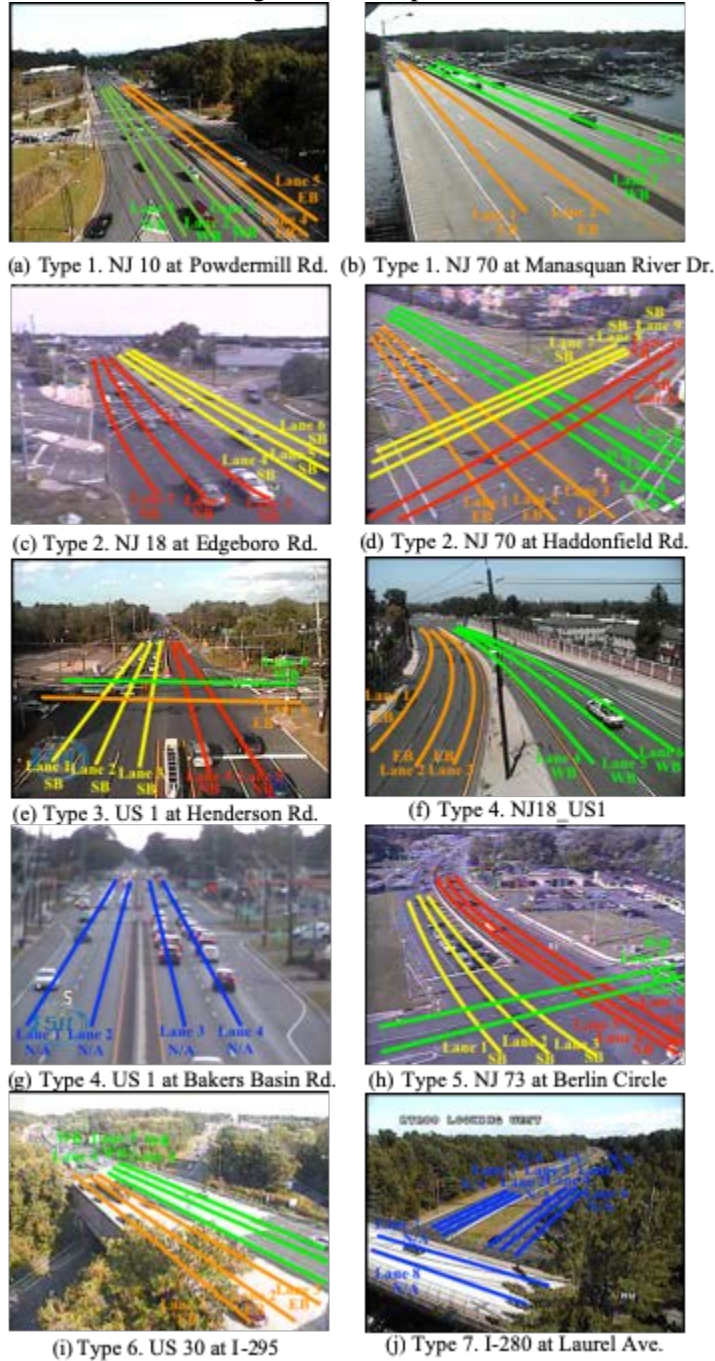


Figure 29 Lane Direction Determination Sample

Figure 29 shows some results of Lane Direction Determination. Figure 29(a)(b) are 2 samples of type 1 cameras that are installed at the roadside. Type 1 cameras that only have 2 directions of lanes are easy to

deal with and the lanes of different directions are marked in different colors. Figure 29(c)(d) are 2 samples of type 2 cameras that are installed at the corner of the intersection. The direction of the lanes in type 2 cameras are also determined well. Figure 29(e) shows a sample of type 3 cameras, which are installed in the median of the intersection. Type 3 cameras require extra effort for lane direction detection during STLine marking. It requires the manual checking result of whether it's facing the intersection or not. By default, type 3 cameras should face the intersections. In the sample above, it's facing the intersection of US 1 at Henderson Rd, combined with this extra information, the directions of the lanes in Figure 29(e) can also be determined. Figure 29(f)(g) are two samples of type 4 cameras, which are installed in the median of a roadway. The proposed lane direction determination method is not applicable to symmetry views, which means type 4 and type 7 cameras cannot use the proposed method directly. More features are required for the determination. Figure 29(f) was determined by the curvature of the road, which can be manually matched with the map. However, Figure 29(g) is not determinable, at least not determinable based on its STLines and relative position. Figure 29(h) is a sample of type 5 cameras, which are installed in-between 2 intersections. It requires manual checking of which intersection the cameras are facing by comparing the constructions in the camera view with Google Street Map. Figure 29(i) is a sample of type 6 cameras, which are installed at the corner of interchanges. Usually, type 6 cameras are similar to type 2 cameras. The only difference is that type 6 cameras may not be able to see the lanes beneath. Figure 29(j) is a sample of type 7, which is installed in the median of interchanges. Its asymmetry installation enables the determination like the roadside camera, but the lack of relative position information disables the determination.

#### 9.4. Traffic Flow Detection Results

The ground truth and auto count results from both the proposed algorithm and HASDA model are as follows.

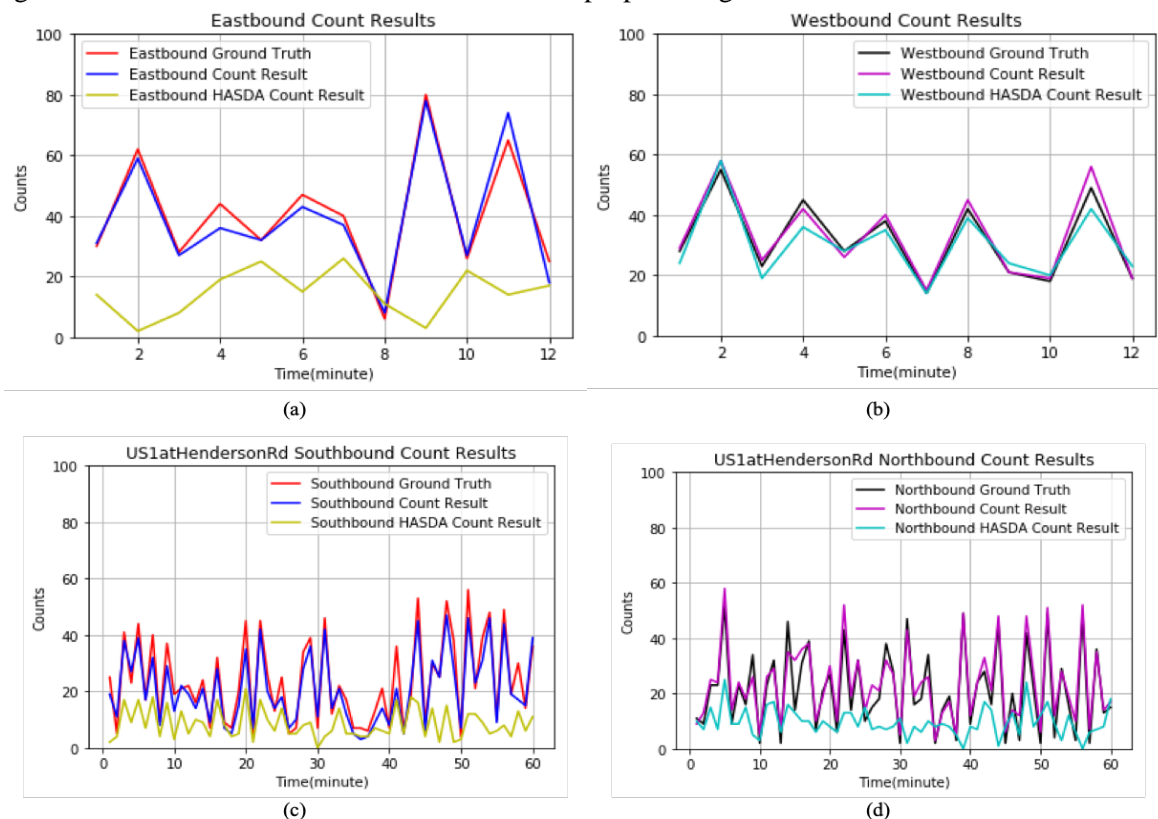


Figure 30 Count Results

Figure 30(a)(b) show the comparison of Ground Truth, Count Result and HASDA Count Result of a 12-min-long video captured from NJ 18 at US 1. Ground Truth is generated from a manual count result completed and checked by experienced human.

From Figure 30(a) which is the comparison of count results in eastbound lanes, HASDA's performance is unsatisfying especially during those minutes with big number of vehicles. HASDA is a model designed for an aerial view, in which there's no occlusion. However, occlusions are quite common in the traffic camera views.

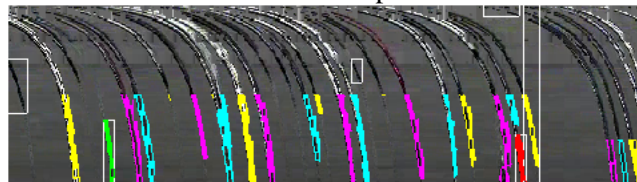
With occlusion detection and removal modules added, the count result of the proposed system has better correlation with ground truth. The count results of 1st, 2nd, 3rd, 5th, 8th, 9th and 10th minute are almost the same and for the results of other minutes, the biggest error is in the 4th minute, when the manual count result is 44 while the auto count result is 36. The reason for undercounts may be the streak, which influenced the strand extraction module of the proposed system and will be discussed in the next section. The ME of the eastbound result generated by the proposed system is 1.25, which means it has 1.25 counts/min on average in ground truth data. The MAE and MAPE of the eastbound result are 3.416 and 10.62%, which is acceptable considering the motorbikes and trucks that may influence the vehicle detection performance and is much better than 26.58 and 59.06% of HASDA.

From Figure 30(b), HASDA has better performance on westbound than eastbound. As has been mentioned before, HASDA is designed for an aerial view, in which most things are symmetry. However, in the traffic camera view, things are no more symmetry. The ME of the westbound result generated by HASDA is 1.5 which means the eastbound result is 1.5 counts/min in the ground truth. The MAE is 3.5, which reflects that on average the HASDA's result in eastbound has 3.5 counts/min difference with ground truth. The MAPE is 11.07%, which means the vehicle detection has around 89% accuracy. The direction determination module enables the proposed system to adjust vehicle detection strategies based on the direction of lanes. In a traffic camera view, vehicles are divided into two types based on their movements: Come to Camera and Away from the Camera. The size and resolution of the vehicles in the camera view vary with the distance, which is quite different from the situation that HASDA has been dealing with and leads to the performance difference of HASDA in processing Come to Camera and Away from Camera STMaps. The ME of the westbound result generated by the proposed system is -1.25, which means it has 1.25 counts/min on average over ground truth. The MAE and MAPE of the westbound result are 2.083 and 5.91%, which means it has 2.08 counts difference on average with ground truth and 94% accuracy of vehicle detection at 1-min-level.

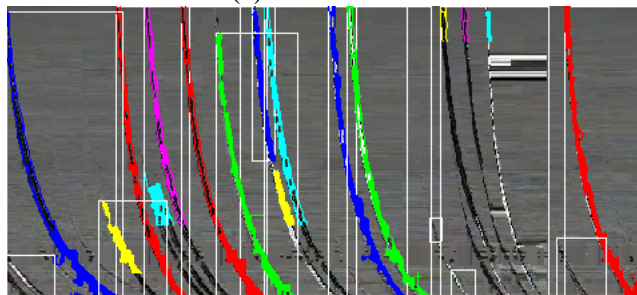
Another test was conducted to evaluate the performance of vehicle counting at intersections at 5-min-level. Figure 30(c)(d) show the result of a 1-hour-long video at the intersection of US 1 and Henderson Rd. This time the proposed system was influenced by the frequent appearance of vans, trucks and crossing vehicles and the ME raise to 6.7 on southbound and -5.2 on northbound, which means that in every 5 minutes there are 6.7 undercounts on southbound and 5.2 overcounts on northbound totally. The undercounts in southbound was caused by the vehicles turned-in which did not get into lanes in time and missed both STLines in neighboring lanes, which will be discussed in the next section. The overcounts in northbound was because the manual count result only counted the vehicles moving out of the intersection, while the proposed system also detected the vehicles joined in from the intersection. There are obvious differences between the manual count result and the auto count result of southbound during the 41st and 58th minute and northbound during 33rd and 46th minute, which was caused by the red light. In both minutes, there were big waves of vehicles waiting for the green light, which makes the occlusion more severe and makes lane-changing happen more frequently. The MAE of the proposed system is around 6.79 for southbound and 6 for northbound, which indicates that in every 5 minutes the number of mis-detected vehicles is around 6. The MAPE is 11.97% for southbound and 11.92% for northbound, which means that the accuracy of vehicle detection is around 88% for the intersection. The result of HASDA has 61% MAPE on southbound and 54% MAPE on northbound, which is much worse than the proposed system.

The proposed algorithm has better performance and stability on videos shot from roadside camera compared with HASDA model. HASDA model has trouble dealing with the occlusion caused by camera angle, which leads to the severe undercounts and has been solved in the proposed algorithm. It also has limitation on removing static noise because of its color-based static noise detection, which leads to severe misdetection

of vehicles because a pole covers the view of WB lanes. In the proposed algorithm, static noise is detected by horizontal edge rather than mean value of color and replaced with clean rows nearby.



(a) Come to camera



(b) Away from camera

Figure 31 Strand Detection Sample

As the figure above shows, Figure 31(a) is the final STMap of a “come to camera” lane and the strands’ endpoints are used to count vehicles because the occlusion usually appears at the start. In a “come to camera” lane, lane changes are dealt by only counting the ones ends in the lane. As can be seen in Figure 31(a), although a lane change threshold of 0.5 was set there are still lane changes that have been overcounted. Figure 31(b) is the final STMap of an “away from camera” lane and the strands’ start points are used to count vehicles. Only the vehicles that start from “away from camera” lane shall be counted to avoid double counting caused by lane changes.

## 9.5. Cloud Deployment and Computational Cost Estimation



Figure 32 CPU Occupancy Sample

HASDA Model took only 91.33s on Intel 8850H@2.6GHz and 111.54s on AWS t2.micro instance while the proposed algorithm took 151.71s to process the 12-min-long video with 5760 frames using one thread on Intel 8850H under Mac OS and 205.68s to process the video on AWS t2.micro instance. The longer processing time is mostly caused by complex occlusion separation and new static noise removal methods

may also have a slight influence. Although the processing time is longer than the HASDA Model, 205.68s for processing a 12-min-long video is still good enough for large-scale cloud deployment.

	LOCATIONSTRING	MISS_CNT	TOTAL_CNT	COMPLETION_RATE
1	I-287 at DMS 3005	2	870	0.9977011494252873563218390804597701149425
2	I-287 at Easton Ave.	5	870	0.9942528735632183908045977011494252873563
3	NJ 20 at US 46	6	870	0.9931034482758620689655172413793103448276
4	I-287 at NJ 10	6	870	0.9931034482758620689655172413793103448276
5	I-287 at US 22 West	6	870	0.9931034482758620689655172413793103448276
6	I-287 at US 22 East	9	870	0.9896551724137931034482758620689655172414
7	I-78 at Garden State Parkway	74	870	0.9149425287356321839080459770114942528736
8	I-78 at NJ 24	550	871	0.3685419058553386911595866819747416762342
9	NJ 36 at NJ 35	733	1084	0.323800738007380073800738007380073800738
10	I-280 East at Garden State Parkway	682	872	0.2178899082568807339449541284403669724771
11	NJ 17 at Orient Way	684	871	0.2146957520091848450057405281285878300804
12	NJ 3 at NJ 17 Camera #1	686	870	0.2114942528735632183908045977011494252874

Figure 33 Performance of Deployment Tests on Sep. 6th, 2019

To further estimate the cost of deployment, an online test was conducted. Three days' cost and workable camera number were collected. On Sep 6th, 7 cameras worked for over 90% of the total time and the cost was \$5.50. On Sep 7th, 15 cameras worked properly, and the cost was \$6.62. On Sep 8th, 12 cameras generated enough results with the cost of \$6.02. On average the cost for each camera is around \$0.5 per day. To process 400 cameras for one year, the total annual cost will be around \$73000.

### 9.6.Limitations of the Proposed Models

The proposed models still have some limitations that need to be addressed in future work. The main limitations include the limitations with lane direction determination for mid-block cameras, vehicle shadows caused by glitches in video streams, low-angle lane or vehicle occlusions, and scanline readjustment for PTZ operations. The detailed descriptions are as follows.

#### 9.6.1.Lane direction determination for Type 4 & Type 7



Figure 34 Lane Direction Determination Sample at US 1 at Bakers Basin Rd.

The proposed lane direction determination model is based on the asymmetry of installation and traffic view. However, there are 217 of 482 NJDOT cameras installed symmetrically or do not have the installation



information which cannot be determined. For example, the camera installed at US 1 at Bakers Basin Rd. is one of the lane-direction-undetermined type 4 cameras. The only solution to match the lanes for now is to check and compare the features such as buildings, forests, billboards, etc.

### 9.6.2. Streak caused missing information in STMaps



(a)



(b)

Figure 35 Streak Samples at 4th min in the tested 12-min video

As Figure 35 shows, the streaks resulted from poor bandwidth connected different vehicles together, which resulted in the severe occlusion and could not be separated by analyzing STMaps with missing information.

### 9.6.3.vehicle coverage caused by low angle



Figure 36 Sample of Coverage by Large Vehicles in Neighboring Lanes

As has been emphasized, the proposed STLine based vehicle detection method saves a lot of computing resources by only extracting the information from STLines, which also results in the loss of information and disables the ability to look outside the STMaps. In Figure 36, limited by the loss of information, what the proposed system can process is only the STMap generated from the information that STLines extracted, which prevents it from detecting the vehicles covered by large vehicles in neighboring lanes even if they show parts for a while that humans can see and recognize them.

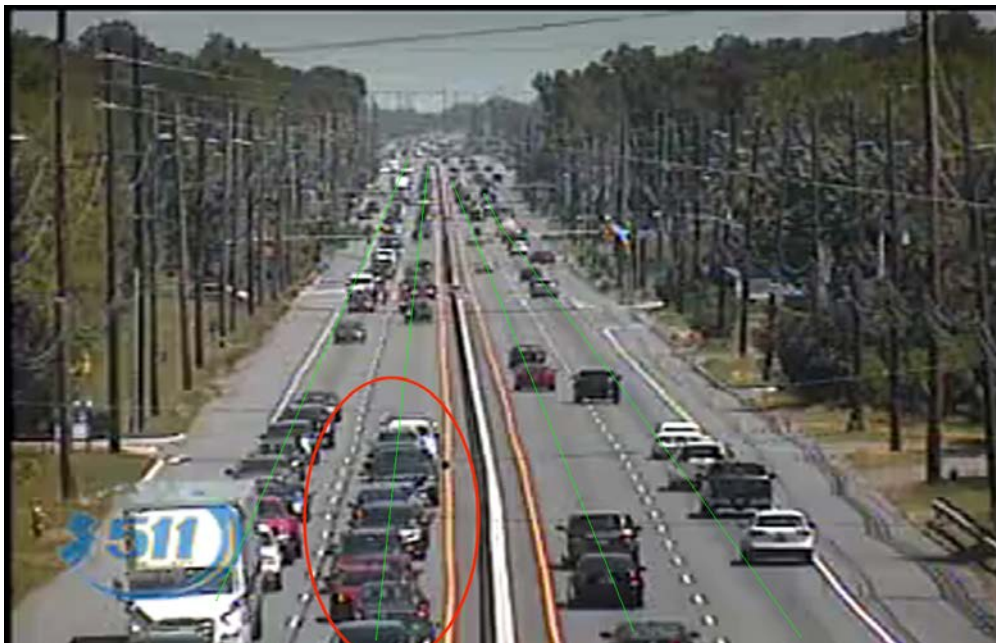


Figure 37 Sample of Coverage by Vehicles in Same Lane

The proposed occlusion detection model relies on the separation that usually happens when the vehicles come close to the camera. However, if the camera angle is too low, there will not be any separation, which directly makes it impossible to deal with occlusions.

#### 9.6.4. PTZ operations caused STLine matching issue



Figure 38 Sample of STLine Shifting Caused by PTZ Operations

Most of the NJDOT cameras support PTZ operations, which will result in the shift of STLines as Figure 38 shows. Then the STLines have to be updated or they will not be able for vehicle counting.

## 10. Cloud Deployment Strategies and Cost Analysis Results

### 10.1. Proposed System Deployment Schematics with Existing TRANSCOM Systems

The following figures show the deployment schematics of the proposed cloud-based traffic counter over the existing TRANSCOM (XCM) system (Consensus Systems Technologies, 2015). The traffic counter takes the real-time CCTV video feed either from through 511NJ or the video servers by Traffic Management Center or TRANSCOM. The traffic count will generate traffic flow data with the same roadway link system used by TRANSCOM to be integrated into their data fusion engine (XCM DFE). Then the flow data will be incorporated into the XCM Data Exchange (XCM DE) and XCM SPATEL and archiving systems for agency data archiving and sharing.

### Private Cloud Deployment Scenery

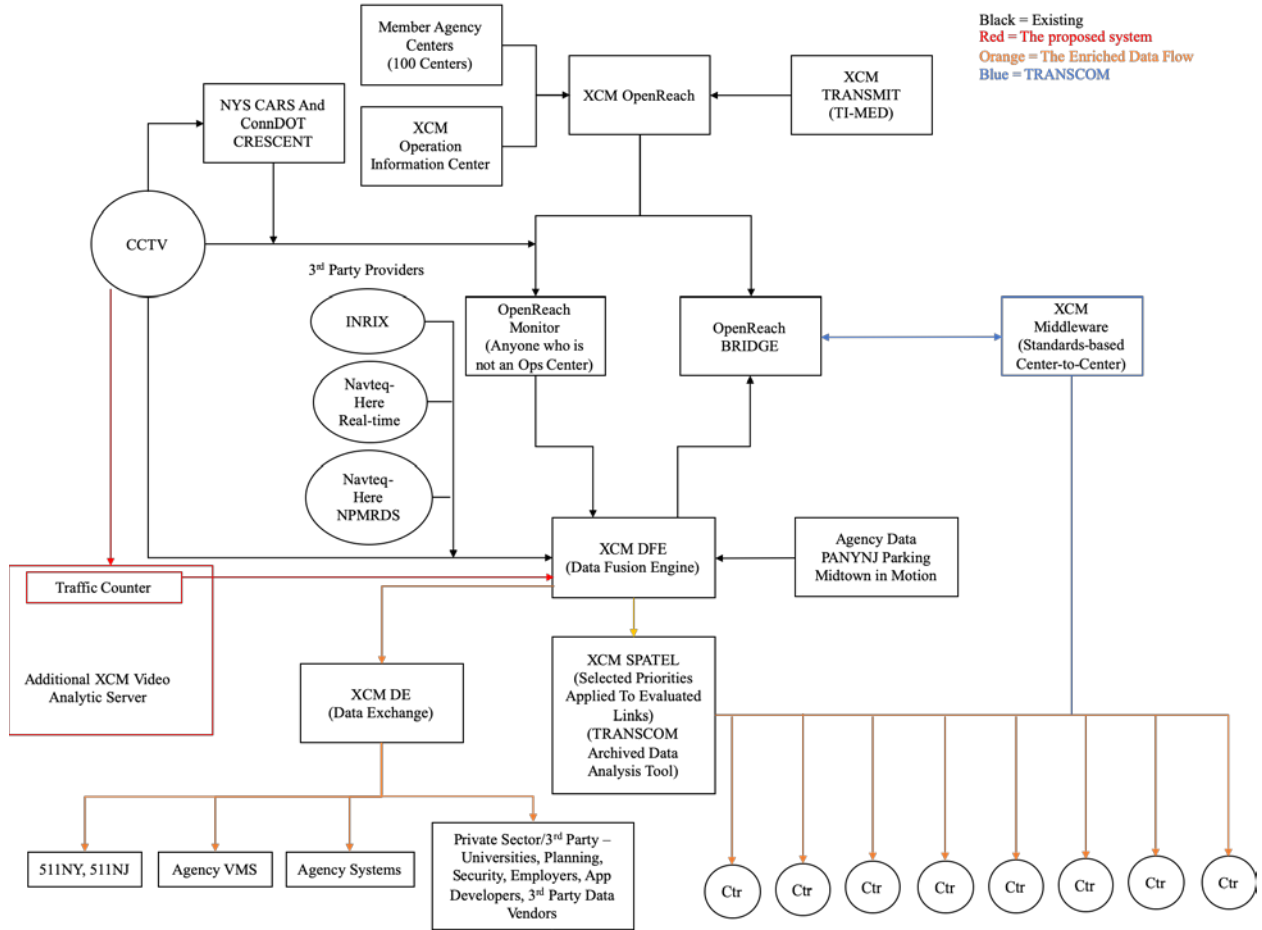


Figure 39 Private Cloud Deployment Schematics with Existing TRANSCOM Systems

### AWS Cloud Deployment Scenery

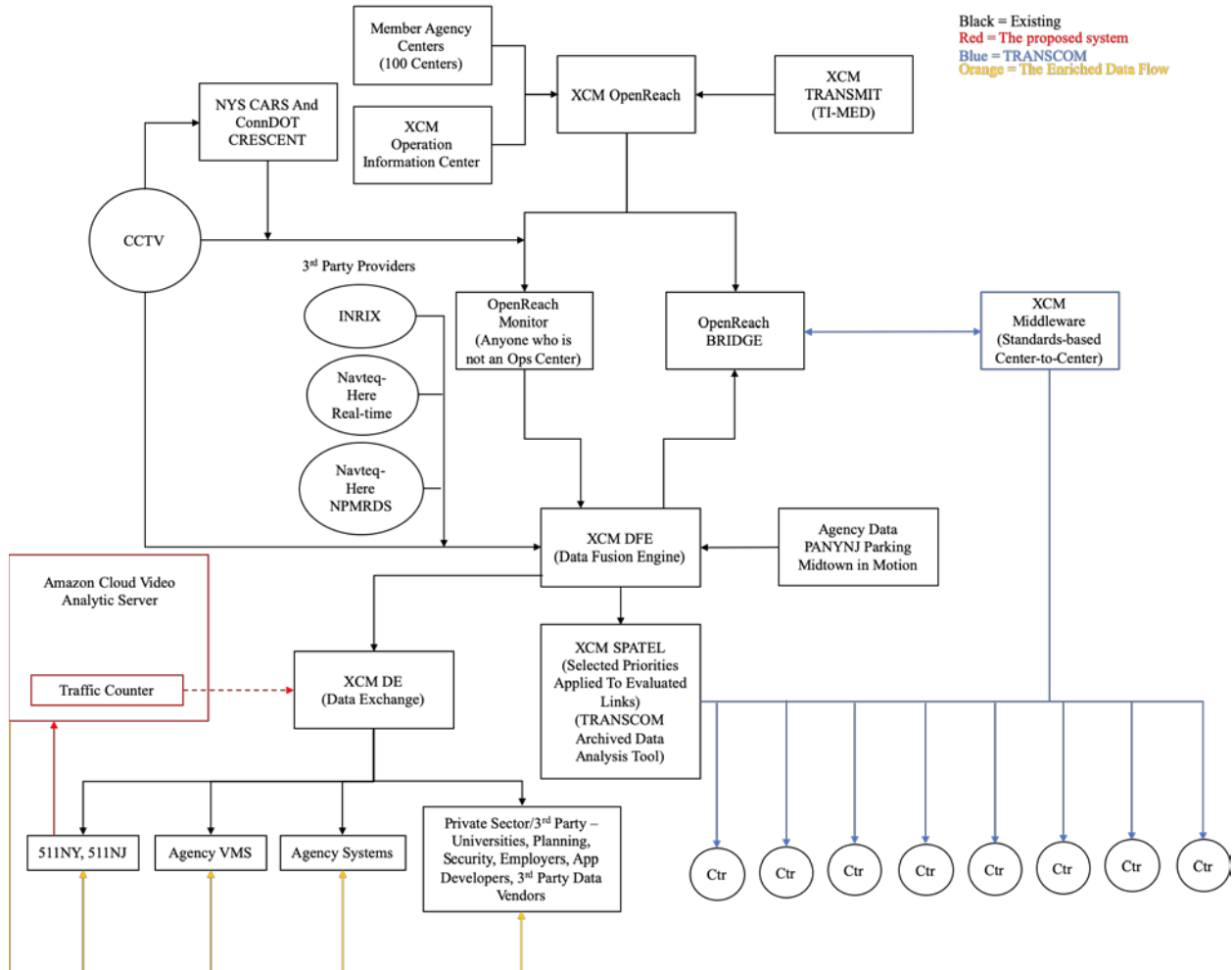


Figure 40 Amazon Cloud Deployment Schematics with Existing TRANSCOM Systems

Traffic counter module can be deployed either on outside third-party cloud like Amazon Web Services (AWS) or be deployed at a dedicated server system within TRANSCOM or NJ Traffic Management Center. If it is deployed at AWS, there are some additional communication and network configuration to access the video stream and transfer the result.

### 10.2 Private versus Commercial Cloud Deployment

Deploying the proposed platform can be deployed both in private cloud in dedicated server clusters within transportation agencies and public cloud services like AWS, Microsoft Azure, or other cloud platforms. The following is a comparison of the pros and cons of using different deployment strategies.

Table 13 Comparison between Commercial Cloud Deployment and Personal Cloud Deployment

	Commercial Cloud	Personal Cloud
Server Types	Cloud Server Instances	Dedicated Servers
Initial Cost	Free	High
Computing Cost	High	Low
Operations/Maintenance	Easy	Complicated
Communication Cost	Free(Usually High)	Low
Storage Cost	Low(Usually High)	Low
Upgrades	Easy and Cheap	Difficult and Expensive
Security	AWS Security	Private
Data Dissemination	Easy and Free(Usually Expensive)	Low

$$C_{Total} = C_{int} + C_{cp} * T_{cp} + C_{tsf} * D_{stg} + C_{hd} * D_{stg}$$

Cloud server instances do not require any initial cost while private cloud on dedicated servers do cost a lot to purchase hardware.

- **Initial Cost:** Cloud server instances do not require any initial cost because they charge for the usage, while dedicated servers require an initial cost for server purchasing and deployment.
- **Computing Cost:** The proposed system requires much computational resource which results in high computing cost for cloud server instances. In the long term, dedicated servers cost less for computing because only power consumption and appropriate maintenance fee will cost after purchase.
- **Operations/Maintenance:** The operation/maintenance of cloud server instances is much easier than dedicated servers as all it needs is to sign in to the console and click the actions. However, the dedicated servers require manual management, security check, periodically backup, etc.
- **Communication Cost:** Benefit from the high-in-low-out network demand of the proposed video analytic system and the output-charge-only policy of AWS, the communication cost of cloud server instances is almost free, while the dedicated servers require high bandwidth to access real-time traffic videos. The low-resolution video streams take up around 800kbps/camera, which means it requires around 320Mbps of download bandwidth to access 400 cameras.
- **Storage Cost:** The Get-Process-Drop design of the traffic counter module reduces the need for storage. Only the output result will be recorded and stored in the database for long term storage, which makes the storage cost very little.
- **Upgrades:** The upgrades of cloud server instances are quite easy without extra hardware purchase fee. All it needs is just to open new instances with better/newer CPUs. However, dedicated servers require a completely upgrade instead, which will be close to the initial cost.
- **Security:** The data of cloud server instances is protected by AWS security, which is reliable enough. The data of dedicated servers is private.
- **Data Dissemination:** As has been mentioned before, the high-in-low-out network demand of the proposed system fit into the free 1GB/month output data policy, which does not cost extra money. Meanwhile, the low bandwidth demand for uploading data does not cost much either.

## 11. Conclusions and Future Work

In this research, we proposed, built, deployed, tested, and evaluated a cloud-based traffic counting system based on CCTV traffic video streams. The proposed system improved some of the existing traffic counting algorithms for low-angle CCTV camera by novel methods to use a fraction of the video frames for analytics (the STLine), efficient processing of static noises caused by roadway infrastructure and signs and occlusion among vehicles. The streamlined workflow of the proposed platform alleviated the limitation and the instability of storage and the modularized system design allows for further improvement to be easily deployed in the future. The proposed system is able to completely support the automatic detection of camera directions with three types of roadside and intersection camera location scenarios and the manual processing of the camera directions with other cameras. Compared with traditional video traffic monitoring systems, the proposed high-efficiency STMap-based system can process real-time video with low consumption of computing and publish the result data feed with slight delay. The detailed contributions are as follows.

- **Video Analytic Models:** To solve the ubiquitous lane matching problem, an installation-asymmetry based lane direction determination method is proposed to match the lanes in camera view to real lanes. Combining the limited information that STLine extracted with the asymmetry in CCTV traffic camera view, the proposed system simplifies the occlusion problem to finding the separation parts of the occlusion in STMap. An occlusion detection method is proposed based on the assumption that most of the occlusions happen in CCTV camera views are far from the camera and will separate from each other when they are close to the cameras. An existing-time-based static noise removal method is proposed to help with background removal in STMap.
- **Adapting for NJ CCTV Traffic and Video Data Sources:** The proposed system combined 511NJ traffic video stream, pre-marked STLine coordinates, camera installation details table, and TRANSCOM link condition data feed together, which enabled the whole Get-Process-Drop process of reading real-time CCTV traffic video streams, generating and processing STMaps, sending the counting results to database, publishing the counting results as feed. The pre-marked STLine coordinates were used to generate STMaps from the video streams. A camera installation detailed table from NJDOT was used to match the cameras and the lanes in camera views with TRANSCOM links. In case that the pre-marked STLines might be ineffective due to PTZ operations, the proposed system also has a local version that can display the video stream and show the pre-marked STLines, which enables the operator to determine whether the STLines are effective and re-mark the STLines if needed.
- **Cloud-based Deployment:** The proposed system including video processing module, database module and feed publishing module was deployed on cloud using AWS RDS DB instances and AWS EC2 instances. An Amazon Machine Image (AMI) sample containing Amazon Linux with the required software and dependency packages was created for easy duplication of the proposed system and can be shared with Amazon Account ID easily. Daily reboot and auto-re-initiation were set to further reduce the cost and improve the reliability.
- **Computational Cost Reduction:** The proposed real-time traffic counter cost much less than traditional systems in the market as it can be deployed on a normal computer with a CPU and network access, not like the traditional systems such as Autoscope, let alone the deep-learning-based traffic monitoring system such as GoodVision. In terms of effectiveness, it also processes existing video much faster than traditional systems.

The proposed system took 151.71s to process the 12-min-long video with 5760 frames using one thread on Intel 8850H under Mac OS and 205.68s to process the video on AWS t2.micro instance. The MAPE were 5%-10%, which were fine considering the low cost it took and could be improved in the future. When dealing with real-time video, the proposed system took up 2%-3% CPU and 100MB RAM per camera on a server with 2 E52470 v2 Xeon CPUs, which proved that the proposed system was quite suitable for large scale deployment.

- **Promising Detection Performance and Efficiency:** Compared the generated results with ground truth, the proposed system has better performance dealing with cameras installed on highway at a medium angle, in which the vehicles get separated from each other. For the occlusion happens at intersections because of red lights, the occlusion detection may not work as expected.

Future work on the proposed platform will be conducted from the following key directions.

- **Automated Detection of Camera Direction:** The proposed methods for camera detection still relies on some knowledge from transportation agencies regarding the relative locations of cameras with respect to the highway or intersections. Future research will focus on the use of existing Google satellite and google street view images to graphically match the video images to determine the precise location of the cameras and traffic directions.
- **Adaptive STLine detection with PTZ operations:** For now, the STLines still need to be marked manually and every time a Pan-Tilt-Zoom (PTZ) operation is performed, the STLines have to be re-marked, which is definitely not a smart choice. There are two potential solutions to solve this problem. 1. Since that the STLines are already marked, what we need to do is realize the PTZ operation detection and once a PTZ operation is performed, do the feature-based auto recalibration to adjust the STLines. 2. Since the STLines should be close to the most frequent trajectory of each lane, the STLines can be marked using traditional trajectory detection. However, these two solutions both require video processing for the whole video view, which is not compatible with the purpose of STLine methods: make it compute easily.
- **Occlusion Removal for CCTV Traffic Cameras:** The proposed separation-detection-based occlusion detection is limited by the assumption that the vehicles will separate from each other when they come close to the camera, which requires the camera installed at the medium or high angle. To solve the occlusions happen near the camera because of large vehicle coverages at low angle cameras or side view angle cameras, color and size of the strands may need to be considered as reference data to separate occlusion.
- **Lane-change Tracking and Processing:** The proposed system has a simple lane change detection which can only correct the vehicle counts rather than track the vehicles in neighboring lanes. To track the whole process of vehicle lane change behavior, it's necessary to create the relationship between neighboring lanes and calibrate the coordinates of neighbor STLines.
- **Deep-learning based Video Analytic Models:** In addition, deep-learning-based image analysis may also be added to analyze the STMap to extract the strands, which is different from a deep-learning-based traffic monitoring system because it will only focus on the extracted STMaps rather than the whole video.

## 12. Appendix

### References

- [1] Bas, E., Tekalp, A. M., & Salman, F. S. (2007, 06). Automatic Vehicle Counting from Video for Traffic Flow Analysis. 2007 IEEE Intelligent Vehicles Symposium. doi:10.1109/ivs.2007.4290146
- [2] Beymer, D., Mclauchlan, P., Coifman, B., & Malik, J. (n.d.). A real-time computer vision system for measuring traffic parameters. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. doi:10.1109/cvpr.1997.609371
- [3] Bharati, P., & Pramanik, A. (2019, 08). Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey. Computational Intelligence in Pattern Recognition Advances in Intelligent Systems and Computing, 657-668. doi:10.1007/978-981-13-9042-5\_56
- [4] Birchfield, S. T., Sarasua, W. A., & Kanhere, N. K. (2010). Computer vision traffic sensor for fixed and pan-tilt-zoom cameras (No. Highway IDEA Project 140).
- [5]
- [6] Canny, J. (1987). A Computational Approach to Edge Detection. Readings in Computer Vision, 184-203. doi:10.1016/b978-0-08-051581-6.50024-6



- [7] Chen, Y., Wu, B., Huang, H., & Fan, C. (2011, 05). A Real-Time Vision System for Nighttime Vehicle Detection and Traffic Surveillance. *IEEE Transactions on Industrial Electronics*, 58(5), 2030-2044. doi:10.1109/tie.2010.2055771
- [8] Cheng, H., Du, H., Hu, L., & Glazier, C. (2005, 01). Vehicle Detection and Classification Using Model-Based and Fuzzy Logic Approaches. *Transportation Research Record: Journal of the Transportation Research Board*, 1935, 154-162. doi:10.3141/1935-18
- [9] Cho, Y., & Rice, J. (2006, 12). Estimating Velocity Fields on a Freeway From Low-Resolution Videos. *IEEE Transactions on Intelligent Transportation Systems*, 7(4), 463-469. doi:10.1109/tits.2006.883934
- [10] Coifman, B., Beymer, D., Mclauchlan, P., & Malik, J. (1998, 08). A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4), 271-288. doi:10.1016/s0968-090x(98)00019-9
- [11] Corovic, A., Ilic, V., Duric, S., Marijan, M., & Pavkovic, B. (2018, 11). The Real-Time Detection of Traffic Participants Using YOLO Algorithm. 2018 26th Telecommunications Forum (TELFOR). doi:10.1109/telfor.2018.8611986
- [12] Cucchiara, R., Piccardi, M., & Mello, P. (n.d.). Image analysis and rule-based reasoning for a traffic monitoring system. *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383)*. doi:10.1109/itsc.1999.821156
- [13] Dailey, D., Cathey, F., & Pumrin, S. (2000, 06). An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, 1(2), 98-107. doi:10.1109/6979.880967
- [14] Dalal, N., & Triggs, B. (n.d.). Histograms of Oriented Gradients for Human Detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). doi:10.1109/cvpr.2005.177
- [15] Dong, Z., Wu, Y., Pei, M., & Jia, Y. (2015, 08). Vehicle Type Classification Using a Semisupervised Convolutional Neural Network. *IEEE Transactions on Intelligent Transportation Systems*, 16(4), 2247-2256. doi:10.1109/tits.2015.2402438
- [16] Fischler, M. A., & Bolles, R. C. (1981, 06). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381-395. doi:10.1145/358669.358692
- [17] Gupte, S., Masoud, O., Martin, R., & Papanikolopoulos, N. (2002, 03). Detection and classification of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 3(1), 37-47. doi:10.1109/6979.994794
- [18] Home. (n.d.). Retrieved from <http://www.autoscope.com/>
- [19] Hsieh, J., Yu, S., Chen, Y., & Hu, W. (2006, 06). Automatic Traffic Surveillance System for Vehicle Tracking and Classification. *IEEE Transactions on Intelligent Transportation Systems*, 7(2), 175-187. doi:10.1109/tits.2006.874722
- [20] Ishak, S. S., Codjoe, J., Mousa, S., Jenkins, S., & Bonnette, J. (2016). Traffic counting using existing video detection cameras. Louisiana Transportation Research Center.
- [21] Iwasaki, Y., Takehara, H., Miyata, T., Kuramoto, T., Kitajima, T., & Setoguchi, M. (2018, 12). Automatic Measurement of Road Traffic Volumes and Vehicle Trajectories Using an Object Detection Algorithm YOLO. *Proceedings of The 6th Virtual Multidisciplinary Conference*. doi:10.18638/quaesti.2018.6.1.387
- [22] Jordan, B., Lennon, W., & Holm, B. (1973, 12). An Improved Algorithm for the Generation of Nonparametric Curves. *IEEE Transactions on Computers*, C-22(12), 1052-1060. doi:10.1109/t-c.1973.223650
- [23] Kaewtrakulpong, P., & Bowden, R. (2002). An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection. *Video-Based Surveillance Systems*, 135-144. doi:10.1007/978-1-4615-0913-4\_11

- [24] Kamijo, S., Matsushita, Y., Ikeuchi, K., & Sakauchi, M. (2000, 06). Traffic monitoring and accident detection at intersections. *IEEE Transactions on Intelligent Transportation Systems*, 1(2), 108-118. doi:10.1109/6979.880968
- [25] Kanhere, N. K., Birchfield, S. T., & Sarasua, W. A. (2008, 01). Automatic Camera Calibration Using Pattern Detection for Vision-Based Speed Sensing. *Transportation Research Record: Journal of the Transportation Research Board*, 2086(1), 30-39. doi:10.3141/2086-04
- [26] Kanhere, N. K., Birchfield, S. T., Sarasua, W. A., & Whitney, T. C. (2007, 01). Real-Time Detection and Tracking of Vehicle Base Fronts for Measuring Traffic Counts and Speeds on Highways. *Transportation Research Record: Journal of the Transportation Research Board*, 1993(1), 155-164. doi:10.3141/1993-21
- [27] Kanhere, N., & Birchfield, S. (2008, 03). Real-Time Incremental Segmentation and Tracking of Vehicles at Low Camera Angles Using Stable Features. *IEEE Transactions on Intelligent Transportation Systems*, 9(1), 148-160. doi:10.1109/tits.2007.911357
- [28] Kilger, M. (n.d.). A shadow handler in a video-based real-time traffic monitoring system. [1992] *Proceedings IEEE Workshop on Applications of Computer Vision*. doi:10.1109/acv.1992.240332
- [29] Koller, D., Weber, J., Huang, T., Malik, J., Ogasawara, G., Rao, B., & Russell, S. (n.d.). Towards robust automatic traffic scene analysis in real-time. *Proceedings of 1994 33rd IEEE Conference on Decision and Control*. doi:10.1109/cdc.1994.411746
- [30] Koller, D., Weber, J., & Malik, J. (1994). Robust multiple car tracking with occlusion reasoning. *Computer Vision — ECCV '94 Lecture Notes in Computer Science*, 189-196. doi:10.1007/3-540-57956-7\_22
- [31] Laureshyn, A., & Nilsson, M. (2018, 06). How Accurately Can We Measure from Video? Practical Considerations and Enhancements of the Camera Calibration Procedure. *Transportation Research Record: Journal of the Transportation Research Board*, 2672(43), 24-33. doi:10.1177/0361198118774194
- [32] Li, Y., Zhu, F., Ai, Y., & Wang, F. (2007, 06). On Automatic and Dynamic Camera Calibration based on Traffic Visual Surveillance. *2007 IEEE Intelligent Vehicles Symposium*. doi:10.1109/ivs.2007.4290140
- [33] Lin, J., & Sun, M. (2018, 11). A YOLO-Based Traffic Counting System. *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. doi:10.1109/taai.2018.00027
- [34] Magee, D. R. (2004, 02). Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 22(2), 143-155. doi:10.1016/s0262-8856(03)00145-8
- [35] Malinovsky, Y., Wu, Y., & Wang, Y. (2009, 01). Video-Based Vehicle Detection and Tracking Using Spatiotemporal Maps. *Transportation Research Record: Journal of the Transportation Research Board*, 2121(1), 81-89. doi:10.3141/2121-09
- [36] Masoud, O., Papanikolopoulos, N., & Kwon, E. (2001, 03). The use of computer vision in monitoring weaving sections. *IEEE Transactions on Intelligent Transportation Systems*, 2(1), 18-25. doi:10.1109/6979.911082
- [37] Michalopoulos, P. (1991, 02). Vehicle detection video through image processing: The Autoscope system. *IEEE Transactions on Vehicular Technology*, 40(1), 21-29. doi:10.1109/25.69968
- [38] Morris, B., & Trivedi, M. (2008, 09). Learning, Modeling, and Classification of Vehicle Track Patterns from Live Video. *IEEE Transactions on Intelligent Transportation Systems*, 9(3), 425-437. doi:10.1109/tits.2008.922970
- [39] Oh, J., Min, J., Kim, M., & Cho, H. (2009, 01). Development of an Automatic Traffic Conflict Detection System Based on Image Tracking Technology. *Transportation Research Record: Journal of the Transportation Research Board*, 2129(1), 45-54. doi:10.3141/2129-06
- [40] Omeroglu, A. N., Kumbasar, N., Oral, E. A., & Ozbek, I. Y. (2019, 04). Mask R-CNN Algoritması ile Hangar Tespiti Hangar Detection with Mask R-CNN Algorithm. *2019 27th Signal Processing and Communications Applications Conference (SIU)*. doi:10.1109/siu.2019.8806552

- [41] Pang, C., Lam, W., & Yung, N. (2007, 09). A Method for Vehicle Count in the Presence of Multiple-Vehicle Occlusions in Traffic Images. *IEEE Transactions on Intelligent Transportation Systems*, 8(3), 441-459. doi:10.1109/tits.2007.902647
- [42] Pang, C., Lam, W., & Yung, N. (2004, 09). A Novel Method for Resolving Vehicle Occlusion in a Monocular Traffic-Image Sequence. *IEEE Transactions on Intelligent Transportation Systems*, 5(3), 129-141. doi:10.1109/tits.2004.833769
- [43] Rad, R., & Jamzad, M. (2005, 07). Real time classification and tracking of multiple vehicles in highways. *Pattern Recognition Letters*, 26(10), 1597-1607. doi:10.1016/j.patrec.2005.01.010
- [44] Rahman, M., Islam, M., Calhoun, J., & Chowdhury, M. (2019, 05). Real-Time Pedestrian Detection Approach with an Efficient Data Communication Bandwidth Strategy. *Transportation Research Record: Journal of the Transportation Research Board*, 2673(6), 129-139. doi:10.1177/0361198119843255
- [45] Ren, S., He, K., Girshick, R., & Sun, J. (2017, 06). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137-1149. doi:10.1109/tpami.2016.2577031
- [46] Schoepflin, T., & Dailey, D. (2003, 06). Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2), 90-98. doi:10.1109/tits.2003.821213
- [47] Schoepflin, T. N., & Dailey, D. J. (2003, 01). Correlation Technique for Estimating Traffic Speed from Cameras. *Transportation Research Record: Journal of the Transportation Research Board*, 1855(1), 66-73. doi:10.3141/1855-08
- [48] Schoepflin, T. N., & Dailey, D. J. (2004, 01). Cross-Correlation Tracking Technique for Extracting Speed from Cameras Under Adverse Conditions. *Transportation Research Record: Journal of the Transportation Research Board*, 1867(1), 36-45. doi:10.3141/1867-05
- [49] Song, K., & Tai, J. (2006, 10). Dynamic Calibration of Pan-Tilt-Zoom Cameras for Traffic Monitoring. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 36(5), 1091-1103. doi:10.1109/tsmcb.2006.872271
- [50] Stauffer, C., & Grimson, W. (n.d.). Adaptive background mixture models for real-time tracking. *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*. doi:10.1109/cvpr.1999.784637
- [51] Stauffer, C., & Grimson, W. (2000). Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 747-757. doi:10.1109/34.868677
- [52] Tai, J., Tseng, S., Lin, C., & Song, K. (2004, 06). Real-time image tracking for automatic traffic monitoring and enforcement applications. *Image and Vision Computing*, 22(6), 485-501. doi:10.1016/j.imavis.2003.12.001
- [53] Unzueta, L., Nieto, M., Cortes, A., Barandiaran, J., Otaegui, O., & Sanchez, P. (2012, 06). Adaptive Multicue Background Subtraction for Robust Vehicle Counting and Classification. *IEEE Transactions on Intelligent Transportation Systems*, 13(2), 527-540. doi:10.1109/tits.2011.2174358
- [54] Veeraraghavan, H., Masoud, O., & Papanikolopoulos, N. (2003, 06). Computer vision algorithms for intersection monitoring. *IEEE Transactions on Intelligent Transportation Systems*, 4(2), 78-89. doi:10.1109/tits.2003.821212
- [55] Zhang, B. (2013, 03). Reliable Classification of Vehicle Types Based on Cascade Classifier Ensembles. *IEEE Transactions on Intelligent Transportation Systems*, 14(1), 322-332. doi:10.1109/tits.2012.2213814
- [56] Zhang, G., Avery, R. P., & Wang, Y. (2007, 01). Video-Based Vehicle Detection and Classification System for Real-Time Traffic Data Collection Using Uncalibrated Video Cameras. *Transportation Research Record: Journal of the Transportation Research Board*, 1993(1), 138-147. doi:10.3141/1993-19
- [57] Zhang, T., & Jin, P. J. (2019, 06). A longitudinal scanline based vehicle trajectory reconstruction method for high-angle traffic video. *Transportation Research Part C: Emerging Technologies*, 103, 104-128. doi:10.1016/j.trc.2019.03.015

- [58] Zhang, W., Wu, Q., Yang, X., & Fang, X. (2008, 03). Multilevel Framework to Detect and Handle Vehicle Occlusion. *IEEE Transactions on Intelligent Transportation Systems*, 9(1), 161-174. doi:10.1109/tits.2008.915647
- [59] Zhang, Z., Tan, T., Huang, K., & Wang, Y. (2013, 03). Practical Camera Calibration From Moving Objects for Traffic Scene Surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(3), 518-533. doi:10.1109/tcsvt.2012.2210670
- [60] Zhao, R., & Wang, X. (2013, 06). Counting Vehicles from Semantic Regions. *IEEE Transactions on Intelligent Transportation Systems*, 14(2), 1016-1022. doi:10.1109/tits.2013.2248001
- [61] Zheng, Y., & Peng, S. (2014, 04). A Practical Roadside Camera Calibration Method Based on Least Squares Optimization. *IEEE Transactions on Intelligent Transportation Systems*, 15(2), 831-843. doi:10.1109/tits.2013.2288353
- [62] Zhou, J., Gao, D., & Zhang, D. (2007, 01). Moving Vehicle Detection for Automatic Traffic Monitoring. *IEEE Transactions on Vehicular Technology*, 56(1), 51-59. doi:10.1109/tvt.2006.883735
- [63] Zhu, Z., Xu, G., Yang, B., Shi, D., & Lin, X. (2000, 07). VISATRAM: A real-time vision system for automatic traffic monitoring. *Image and Vision Computing*, 18(10), 781-794. doi:10.1016/s0262-8856(99)00046-3
- [64] Autoscope. Image Sensing Systems, Inc. [www.autoscope.com](http://www.autoscope.com)
- [65] Citilog, [www.citilog.com](http://www.citilog.com)
- [66] TrafficVision, [www.trafficvision.com](http://www.trafficvision.com)
- [67] GRIDSMART, [www.gridsmart.com](http://www.gridsmart.com)
- [68] AgentVi, [www.agentvi.com](http://www.agentvi.com)
- [69] intruVision, [www.intuvisiontech.com](http://www.intuvisiontech.com)
- [70] SVS, [www.smartcctvltd.com](http://www.smartcctvltd.com)
- [71] Placemeter, [www.placemeter.com](http://www.placemeter.com)
- [72] Traficon, [www.traficon.com](http://www.traficon.com)
- [73] MetroTech, <http://metrotech-net.com>
- [74] GoodVision, <https://goodvisionlive.com>
- [75] Miovision, <https://miovision.com>
- [76] Genetec, <https://www.genetec.com/solutions/all-products/omnicast/kiwivision-video-analytics>
- [77] Aventura, <https://www.aventurasecurity.com>
- [78] TRANSCOM System Concept of Operations, <http://www.infosenseglobal.com/wp-content/uploads/2018/04/transcom-systems.pdf>