Real-time Traffic Estimation at Vehicular Edge Nodes

Gorkem Kar
WINLAB, Rutgers University
gkar87@winlab.rutgers.edu

Shubham Jain
WINLAB, Rutgers University
shubhamj@winlab.rutgers.edu

Marco Gruteser
WINLAB, Rutgers University
gruteser@winlab.rutgers.edu

Fan Bai
General Motors Research
fan.bai@gm.com

Ramesh Govindan
University of Southern California
ramesh@usc.edu

ABSTRACT

Traffic estimation has been a long-studied problem, but prior work has mostly provided coarse estimates over large areas. This work proposes effective fine-grained traffic volume estimation using in-vehicle dashboard mounted cameras. Existing work on traffic estimation relies on static traffic cameras that are usually deployed at crowded intersections and at some traffic lights. For streets with no traffic cameras, some well-known navigation apps (e.g., Google Maps, Waze) are often used to get the traffic information but these applications depend on limited number of GPS traces to estimate speed, and therefore may not show the average speed experienced by every vehicle. Moreover, they do not give any information about the number of vehicles traveling on the road. In this work, we focus on harvesting vehicles as edge compute nodes, focusing on sensing and interpretation of traffic from live video streams. With this goal, we consider a system that uses the dash-cam video collected on a drive, and executes object detection and identification techniques on this data to detect and count vehicles. We use image processing techniques to estimate the lane of traveling and speed of vehicles in real-time. To evaluate this system, we recorded several trips on a major highway and a university road. The results show that vehicle count accuracy depends on traffic conditions heavily but even during the peak hours, we achieve more than 90% counting accuracy for the vehicles traveling in the left most lane. For the detected vehicles, results show that our speed estimation gives less than 10% error across diverse roads and traffic conditions, and over 91% lane estimation accuracy for vehicles traveling in the left most lane (i.e., the passing lane).

CCS CONCEPTS

•**Information systems** →*Mobile information processing systems;*
•**Computer systems organization** →*Real-time system architecture;*

KEYWORDS

Object detection, Traffic estimation, Vehicular sensing, Camera

SEC '17, San Jose / Silicon Valley, CA, USA

DOI: 10.1145/3132211.3134461

ACM Reference format:
Gorkem Kar, Shubham Jain, Marco Gruteser, Fan Bai, and Ramesh Govindan. 2017. Real-time Traffic Estimation at Vehicular Edge Nodes. In *Proceedings of SEC '17, San Jose / Silicon Valley, CA, USA, October 12–14, 2017,* 12 pages.
DOI: 10.1145/3132211.3134461

1 INTRODUCTION

With the evolution of technology, vehicles are becoming increasingly connected and automated. They have evolved into rich sensing platforms with a plethora of diverse sensors. While the stream of sensor data can be communicated to and processed in a remote cloud, bandwidth and latency challenges encourage processing of this data near the edge and on the vehicles themselves.

Traffic estimation. One sample application to use this data is to estimate the traffic. Existing work on traffic estimation relies on traffic surveillance cameras or the GPS-based speed estimation used by navigation apps. However, most roads are not covered by traffic cameras, and GPS-based works estimate the speed of a few users that share their location information with the server which might belong to outliers. Also existing GPS-based approach can only determine the overall direction-level information; but in many cases, lane-level traffic information is critical for both navigation and futuristic autonomous driving features. Therefore, a new method that could be deployed widely and give better speed estimation results is needed.

With high computing power and less power constraints, vehicles provide plentiful opportunities to sense the dynamic environment. We propose to use vehicles as edge compute nodes, focusing on sensing and interpretation of traffic from live video streams. Unlike smartphones, that are constrained in compute resources and available power, vehicles can support efficient compute platforms without the constraints of a small form factor compute node. Additionally, they provide wide reach into remote areas, where other platforms may be unavailable.

With the help of the front facing cameras that are installed in vehicles, we propose to record the traffic and count the vehicles that are traveling. Further, the average speed experienced by each driver can be used to estimate the traffic. Under free-flow traffic conditions drivers have the flexibility to choose higher speeds. However, when the density of vehicles increases, vehicle speeds tend to decrease. Cameras allow capturing this information from many surrounding vehicles, and often many oncoming vehicles. They can therefore gather rich data about traffic conditions.

Existing work. Well-known navigation applications such as Google maps or Waze use the GPS traces of some users, that use

these apps on road, to calculate the speed of these vehicles to estimate the overall traffic conditions. However, since these apps are not used by every single driver on road, the calculated speed belongs only to the drivers that report GPS traces, potentially introducing systematic bias due to uneven sampling. For instance, the calculated speed could belong to a speeding driver in the left lane or a cautious, slow driver in the right lane. Therefore, the estimated traffic information may not be accurate. By using traffic cameras that are deployed on roads more complete traffic data could be obtained at one location. The number of vehicles traveling, their speeds, and congestion on a given road, are just a few questions that could be answered using traffic cameras and have been investigated previously [38], [27], [33], [14] and [10]. However, the number of deployed traffic cameras is not sufficient to cover all roads and vehicles, especially in remote areas.

This work seeks to overcome these challenges with a collaborative sensing system, with vehicle detection, vehicle tracking and traffic estimation components, as shown in Figure 1, by leveraging a dash-cam mounted in vehicle and a processor to process the video stream. The vehicle detection component of such a system could work continuously to detect vehicles and determine bounding-boxes around each vehicle. The vehicle tracking component then tracks the movement of each detected vehicle. With the traffic estimation component, we can then count the number of vehicles on roads, estimate the lane in which they are traveling and their speeds using image processing techniques.

The salient contributions of this work are summarized below.

- An automated traffic estimation framework that manages vehicle detection, vehicle tracking and traffic estimation using a dashboard camera that can achieve wide coverage at low cost.
- We evaluate through multiple days of roadway experiments on a campus road and a major highway, and show it is possible to count the vehicles that are traveling on a given road and determine their speeds. Our system can detect about 90% of vehicles traveling in the left most lane, estimate their speeds with about 10% error.
- Lane estimation for vehicles in the camera's view. Our system achieves more than 91% accuracy in lane estimation for the vehicles that are traveling in the left most lane, a.k.a passing lane.

2 RELATED WORK

There has been much work about vehicle detection and tracking. For vehicle detection, most works [10, 14, 26, 27, 33, 38] assume that the camera is static and vehicles are detected by finding the differences of the images for that camera. Using well-known background subtraction techniques, the only moving objects, vehicles, have been identified and speed estimations are made. Zhu et al [38] and Jung et al [27] have attempted to calibrate the traffic camera using scene information for a particular camera and managed to count vehicles and estimate their speeds. Other works [14, 33] have extended this idea and made it possible to cover any stable camera by first calculating the relative position of the traffic camera to vehicles, then estimating the lane boundaries and finally calculating the mean vehicle speed for each lane. Beymer et al. [12] proposed to use corner features to estimate the traffic flow. Hsu et al. [23] propose to use entropy to estimate vehicle speeds.

Cameras are not the only sensors to detect and track vehicles. Sonar and camera are being used at the same time [28], which achieves vehicle detection at close distances (i.e., sonar distance). Bruzzone et al [13] show that using multiple sensors provides better accuracy in object detection.

In computer vision, for object detection a set of robust features (SIFT [29], convolutional [17] etc.) from images is calculated and then classifiers are used to identify objects. Classification is performed by using a sliding window on some parts of the image. This strategy has been used in many projects [15, 19, 35, 36].

Recently, the YOLO framework [32] created a single convolutional network that can detect multiple objects in an image. It requires a training phase at initialization to work on full images and object coordinates. Then it can process the entire image, without a need for sliding window, and provide relatively accurate object detection, almost in real-time (with latency of 25 ms).

In the realm of object tracking, Xiang et al [37] proposed a multi object tracking framework based on Markov decision processes (MDP). They have two stages: first they collect ground truth trajectories of pedestrians, then a second learning method takes place as decision process is performed by current status and history of the target. At every step, MDP attempts to track the target pedestrian and collects feedback from the ground truth. Then a similarity function is updated with the feedback. The authors manage to track pedestrians 7% better than the second best tracker.

In another work [18], the authors proposed a framework to estimate trajectories of nearby vehicles using four cameras placed diagonally on the car. They modify the MDP tracker that's also being used in [37] to track vehicles. Since the movement of vehicles is not as random as pedestrians, the tracking performance is much better than in the previous effort. With 4 cameras, the trajectory recall is over 90%.

Lane estimation and tracking have been investigated earlier [24, 30]. In these works, the camera, LIDAR, and GPS sensors are used to extract road features such as lane markers and road curvatures, to enable applications such as a lane departure warning system and a driver attention monitoring system. However, only the lane that the camera vehicle is traveling in, is estimated, not those for other vehicles.

3 BACKGROUND AND APPLICATIONS

With rising traffic congestion, many applications may benefit from an accurate estimation of traffic on a particular road. Let us consider the following examples.

Traffic Flow Terms. Traffic can be represented in several terms. Traffic flow represents the number of vehicles that are passing a reference point per unit time (e.g., vehicles per hour). Traffic density represents the number of vehicles per unit distance along the road. The higher either number is, the more congested the road becomes. As roads become more congested, vehicle speed decreases. There exists a well-known relationship between traffic congestion and vehicle speed.

For ease of interpretation, traffic congestion is often represented through a set of discrete Levels of Service (LOS). Table 1 [22] shows

this quantization and the relationship between the foregoing parameters. In this table, level A represents free-flow traffic while level F represents congestion. The unit for Traffic flow is vehicle/hour/lane and the unit for the density is vehicle/mile. Knowing the average speed of vehicles traveling on the road or the traffic-flow, LOS could be determined simply through a table lookup. By using the same table, it is possible to calculate how many vehicles are traveling per mile (i.e., traffic density on the road).

LOS	Speed Range	Flow Range	Density Range
A	Over 60	Under 700	Under 12
B	57-60	700-1100	12-20
C	54-57	1100-1550	20-30
D	46-54	1550-1850	30-42
E	30-46	1850-2000	42-67
F	Under 30	Over 2000	Over 67

Table 1: Levels of Service of a road

The traffic flow and density are direct metrics that show the congestion of roads, or simply the traffic. With higher traffic flow and density numbers, one might expect heavier traffic on roads.

Real-time Car Mapping. With the deployment of DSRC systems, vehicles have the capability to communicate with each other and learn the positions of nearby vehicles. However, for older cars that do not support DSRC, their location would not be known by other vehicles. Some newer vehicles come with built-in GPS receivers but since they don't transmit that information, nearby vehicles are not aware of the location of these cars. Access to a fine-grained traffic estimation system creates awareness of a driver's surroundings by mapping cars in real-time.

Rear-End Collision Prevention. Rear-end collisions are the most common traffic accident in the United States [1]. Vehicles traveling in close proximity to the vehicle in front usually cannot stop in time if the vehicle in front needs to stop suddenly. With DSRC, such accidents are expected to decrease because each vehicle supports DSRC, transmits their location and speed in real time. Being able to stop, of course, depends on the speed of other vehicles and the following distance. The earlier a stopped vehicle is detected, the more time a following driver has to stop in time. A stand-alone speed estimation system on each car, that does not rely on technology that is unavailable on all vehicles, is the need of the hour. A continuous speed estimation of the vehicles around a car, can prevent many mishaps, such as rear-end collision prevention.

4 VEHICULAR EDGE NODES

Vehicles have evolved from mechanical systems to cyber physical systems, generating large amounts of real-time data. They offer high compute capabilities with far less power consumption concerns compared to other mobile platforms. Vehicles are powerhouses of energy, traversing through our physical world, and with the many sensors built in to them they are capable of sensing our dynamic environments.

Vehicles are increasingly being installed with front facing cameras. Originally, these cameras were intended for a specific purpose such as lane detection, lane keeping, and evidence in case of theft or vandalism. However, recently, with the trend in autonomous driving, dashboard mounted cameras have been used for applications ranging from simple pedestrian/car detection [11, 16, 20, 21] to enabling a self-driving system [31, 34]. In addition to enabling vehicle specific or driver specific services, these cameras can be leveraged for large-scale traffic analytics. Cameras in each vehicle have a unique perspective of the observed environment. For example, a car driving in the left most lane has a clear view of the cars driving in the same direction, as well as those in the opposite direction. Similarly, a car in the rightmost lane is optimally placed for detecting stalled vehicles. Each car can be enabled to process raw video streams and compute high level semantic information. Pre-processing raw video streams to extract high level information optimizes bandwidth usage, reduces latency, and conserves privacy.

This information can then be shared with neighboring vehicles or a centrally located map service. The cloud-based map service can aggregate the information from a large number of vehicles to provide an up-to-date map of the region, overlaid with precise traffic and other road conditions information. This constitutes a more accurate and sophisticated assessment of regions, compared to other approaches such as surveillance cameras, that do not cover all areas. Such near real-time fine grained traffic analytics can enhance traffic flow and regulations, optimize transportation, and further assist in provisioning city services.

Using vehicles as edge compute platforms has become possible because of the increasingly powerful computing resources that are becoming available from multiple vendors.[1] These energy efficient compute platforms bring cutting-edge processors and accelerators to cars, enabling sophisticated and very deep networks to process rich video data in near real-time. As we bridge the gap in hardware, this work aims to demonstrate continuous large-scale traffic volume estimation techniques on distributed compute nodes, such as those in vehicles, to demonstrate that such compute resources are not just valuable for advanced driver assistance and automated driving systems, but could also support a plethora of (potentially third-party) analytics applications if the platform becomes more openly programmable.

5 SYSTEM OVERVIEW

To leverage the computation capabilities of vehicular edge nodes and to demonstrate their potential, we design and implement a traffic estimation system. Our goal is to detect and count vehicles, estimate the lanes they are traveling in and calculate the speed of each vehicle. The system consists of three main components: *Vehicle Detection*, *Vehicle Tracking* and *Traffic Estimation* as depicted in the Figure 1. The Vehicle Detection module aims to detect all vehicles in the camera's field-of-view, in real-time. A bounding box is generated for each vehicle, which is then used for tracking the vehicle along its trajectory in the car's view. With simple parking lot experiments, we could detect up-to 6 vehicles in one frame. For real road tests, we could detect up-to 5 vehicles traveling in both directions. Once a vehicle is detected, the *vehicle tracking* module extracts Scale Invariant Feature Transform (SIFT) feature descriptors within each vehicle's bounding box. A vehicle is tracked by matching these feature descriptors between consecutive frames.

[1] A well-publicized example is the NVIDIA DrivePX 2 [4] platform.

Algorithm 1 Vehicle Count Estimation

```
function unique_vehicle_detection(f)   ▷ record frames f of vehicles traveling in the opposite direction
    for each consecutive frames f_i and f_{i+1} do
        N=count_sift_features(f_i, f_{(i+1)});
                                        ▷ Compare with the threshold
        if N > 15 then
            unique(i)=false;
        else
            unique(i)=true;
        end if
    end for
end function
function Increase Count(unique) ▷ Increase the count if the unique vehicle is being seen minimum 5 times
    count=count ++
end function
```

To improve the confidence associated with each detection, we focus on five consecutive frames. If the same vehicle is identified in at least five consecutive frames, we recognize it to be the same vehicle. The *traffic estimation* module, firstly, counts the vehicles in the opposite lane by identifying unique cars based on frame to frame feature tracking. Secondly, it estimates the lane that each car is traveling in, by creating pseudo-lane markers. Thirdly, depending on the lane that car is traveling in, we estimate the speed as shown in Algorithm 3. Each component will be discussed in detail in the next subsection.

For each frame, *vehicle detection* component outputs bounding boxes around the detected vehicle(s). Since there could be multiple bounding boxes, we always use the left most bounding box of the left of the screen to identify the vehicle in the opposite lane. All vehicles detected in the right half of the screen are traveling in the same direction as the car with the dash-cam.

Ideally, a vehicle traveling in the opposite direction would be detected in multiple consecutive frames. In order to get an accurate vehicle count on roads, we need to identify each unique vehicle. This can be achieved by finding similarity for the vehicles in consecutive frames. With the *vehicle tracking* component, we compare SIFT features of the vehicles for those frames. If we have a high number of matched feature points, those two vehicles should be the same one. After analyzing the frames, we observed that for the same vehicle in consecutive frames, there are minimum 15 matched features. So we compare the feature points from different frames and if the number of matching features is greater than 15, those frames correspond to the same vehicle and we do not increase the count. Finally, we check if the object is identified 5 times. For some non-vehicle objects, our algorithm classifies them as vehicle. We exclude them by checking if we have minimum 15 matched feature for 5 consecutive frames. This algorithm is summarized in Algorithm 1.

One can claim that not all the feature points are coming from the vehicle, but also the outside world, such as road segment or trees. The feature point distinction is discussed in *traffic estimation*.

In *traffic estimation* component, we can first calculate the number of vehicles that are traveling by using the vehicle tracking results.

Algorithm 2 Traveling Lane Estimation

```
function Initialize(r)   ▷ We manually record the trajectories of two vehicles from each lane and note down the center coordinates for each bounding box for road r and calculate the line equations for each lane
    ll_flow_line = a_1 . x + b_1
    ml_flow_line = a_2 . x + b_2
    rl_flow_line = a_3 . x + b_3
end function
function Pseudo Lane Marker Generation(r)   ▷ for a particular road r, record frames f of vehicles traveling in the opposite direction
    Initialize(r)   ▷ We calculate the lane markers
    ll_marker = (a_1+a_2)/2 . x + (b_1+b_2)/2
    rl_marker = (a_2+a_3)/2 . x + (b_2+b_3)/2
end function
function Estimate the Lane(bb)   ▷ Compare the center coordinates coord of the bounding box bb with the pseudo-lane markers
    if coord > ll_marker then
        left_lane_vehicles=left_lane_vehicles ++
    else if rl_marker < coord < ll_marker then
        middle_lane_vehicles=middle_lane_vehicles ++
    else
        right_lane_vehicles=right_lane_vehicles ++
    end if
end function
```

For each unique vehicle, we increase the total count. The second step would be to estimate the lane of travel for each detected vehicle. For the vehicles that are traveling in the same direction, we propose to use Hough lines [3], to extract line segments based on Hough transform. In this way, we detect the lines in the road and identify each lane separately. For vehicles traveling in the opposite lane, estimating the traveling lane is harder since we may not always observe the lane markers. We propose to create pseudo-lane markers in the opposite direction and estimate the traveling vehicle lane using those markers. The process is summarized in Algorithm 2.

In order to estimate the speed of each vehicle, we need to know how far that vehicle has moved for consecutive frames. We first calculate the distance change of the matched SIFT feature points in real world for consecutive frames. For this, the camera should be calibrated and this will be discussed in Section 6.4. In this way, we can calculate how each feature point is moved in real world as shown in *distanceCalculation* function in Algorithm 3. However, not all the feature points may belong to the vehicle Therefore, some points may move differently for consecutive frames and the relative distance change for these points should be avoided when calculating the speed. This is the *detectDistanceAnomaly* function. And finally, by using the change of distance of the feature points of the vehicle, we can calculate the speed of that vehicle as shown in *Speed Estimation* function in Algorithm 3.

Our system does not require user interaction since it can automatically detect vehicles and estimate the speed of vehicles. The only exception is to estimate the lane for the opposite side traffic.

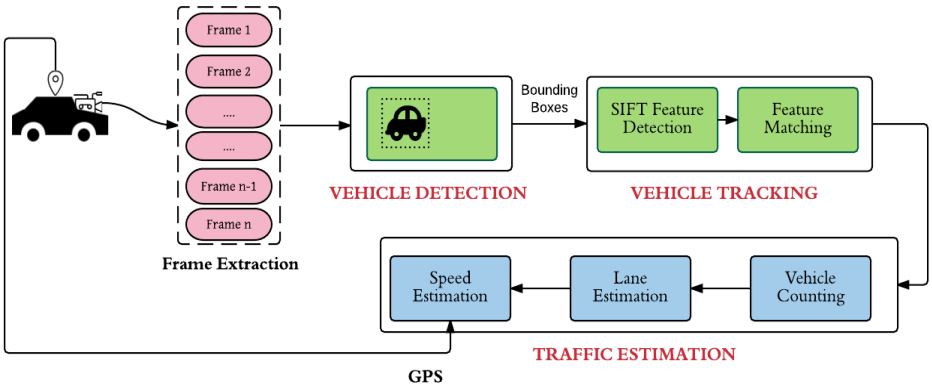


Figure 1: System Overview.

Algorithm 3 Vehicle Speed Estimation

function DISTANCECALCULATION(L, f) ▷ record frames f of vehicles traveling in the opposite direction, L is the horizontal distance between the vehicle and the camera
  for each consecutive frames f_i and f_{i+1} **do**
    $x_{i,j}, x_{i+1,j}$=$get\_sift\_features(f_i, f_{i+1})$;
      ▷ Calculate the path distance
    for each SIFT feature j **do**
      d_j=$calculate\_path\_distance(L, x_{i,j}, x_{i+1,j})$;
    end for
    $m(d)$=$average(d_j)$
    $\sigma(d)$=$std\_dev(d_j)$
  end for
end function
function DETECTDISTANCEANOMALY($m(d), \sigma(d), d$)
  for each SIFT feature j **do**
    if $m(d) - \sigma(d) < d_j < m(d)+\sigma(d)$ **then**
      $valid_d$=$valid_d \cup d_j$
    end if
  end for
end function
function SPEED ESTIMATION($valid_d, fps$)
  $Speed$=$valid_d$xfps
end function

For each road, we need the *Initialize* step in Algorithm 2 to create lane markers for that road, once.

6 TRAFFIC ESTIMATION

In this section, we describe how to detect vehicles on roads, calculate the speed, and lane of travel information to estimate traffic. While we discuss this in the context of vision, similar methods could be applied to other vehicle detection methods such as LIDAR.

6.1 Vehicle Detection

To detect vehicles on road in real-time, we need an object detection framework that can detect vehicles regardless of make, model or color. The detection system must be robust, and resilient to perspective. Prior detection systems often use hand-tuned features to recognize objects in a frame. Recently, convolutional neural networks (CNN) have been proven to perform better than traditional object recognition frameworks. We use a state of the art CNN-based object detection framework called YOLO [32], for detecting vehicles. The network uses features from the entire image to predict objects and marks bounding-boxes around them. With this method, the image is divided into grids and in each grid cell, predictions and confidence scores are generated for different objects. The higher the confidence score, the likelihood of correct object detection increases. YOLO is designed as a convolutional neural network: the initial layers are responsible for extracting features and the connected layers are responsible for predicting the objects with confidence scores. It is extremely fast and streaming videos can be processed with less than 25 ms of latency per frame. Unlike classifier-based approaches, YOLO directly corresponds to detection performance and the entire model is trained jointly. The pre-trained models have been trained for 20 different objects. Since it is designed to identify a wide range of objects, the car detection performance is not satisfactory. We extend the provided car dataset with publicly available car datasets by Stanford University [6] and University of Illinois [7].

6.1.1 Training. With the extended dataset, we are using more than 8000 images of vehicles that are taken from every angle, to train the network. The training was done in a server with a GPU, Quadro K-5000 [2]. The training stage requires labeled images, where the images have a bounding box for each vehicle. The training takes about 16 hours for 8000 images, and is a one-time process. At the end of the training process, a weight file is generated. We use this weight file with the YOLO network to detect vehicles in real-time. We also need to note that in the dataset, we are using the images of cars including sedans, SUVs, coupes, wagons with different colors and years. However, we don't include the images of trucks or buses. Therefore, the detection performance for those vehicles would not be as high as personal cars. That said, YOLO can be trained with a larger dataset to include all types of vehicles.

Figure 2: Detected vehicle.

6.1.2 Testing. For testing the accuracy of vehicle detection, we use a dashboard mounted camera and conduct a small set of experiments in a parking lot with multiple vehicles. The camera is set to record at 30 frames per second, and video recordings lasting a few minutes were made. The primary objective is to detect vehicles in most of the frames. From our tests, we observed that the detection range is about 100 meters. For distances that are larger, the vehicle is not always detected. In Figure 2, a detected vehicle has been shown on the road. With extensive tests that were done both in parking lot and on roads, we observed that vehicles are detected regardless of model, color or type. In the night time, the performance of vehicle detection suffers, due to the lack of night time vehicle images in the training set. The detailed results of vehicle detection in various environments will be discussed in Section 7.

6.2 Vehicle Counting

We consider a system that counts the number of vehicles in the car's field of view. These vehicles could be traveling in the same direction as the camera instrumented car, or in the opposite direction. While traveling, the number of encountered vehicles in the same direction is usually not relevant since we either detect the same vehicles (e.g., our car is following them with constant speed or in a traffic jam) or we keep seeing new vehicles (e.g., we are passing vehicles) or don't see any vehicles (e.g., there is no car traveling in the detection range).

For vehicles traveling in the opposite direction, the camera instrumented car encounters every vehicle, even for a small duration. The challenge however, lies in the detection of these vehicles due to higher relative speeds. Specially at a highway, with average speeds of 55-70 mph, vehicles traveling in the opposite direction make fleeting appearances in the camera frame. Additionally, they are often partially obstructed in the camera's view due to the presence of other vehicles. A counting algorithm must account for all these factors.

The vehicle detection framework described in the previous section can provide bounding boxes for cars in each frame. This gives us an estimate of how many vehicles exist in the frame, and works very well for scenes with static cars, such as our parking lot experiment. However, on real roads, one needs to detect moving vehicles with a camera in motion, which raises severe challenges.

While operating the counting system in-the-wild, on highways, a major challenge is to avoid counting the same vehicle multiple times as it is detected in consecutive frames. To overcome this, we augment our detection with a vehicle tracking module to keep track of vehicles that have been seen and accounted for in previous frames. We use the detections provided by the vehicle detection module, and extract Scale Invariant Feature Transform (SIFT) from the bounding box of each vehicle in each frame. By matching features between consecutive frames, we distinguish new cars in the frame from those that have been sighted before.

Instead of calculating SIFT features over the entire frame, we decide to use the leftmost bounding box in each frame. We first check if the left most bounding box is in the left half of the screen. If not, that vehicle is traveling in the same direction as us and we don't include it in the count. After making sure that vehicle is indeed traveling in the opposite direction, we compare the SIFT features of the objects in the bounding boxes for consecutive frame as discussed in Algorithm 1. If a vehicle is detected for 5 consecutive frames, we increase the count.

SIFT features of consecutive frames are calculated using VLFeat library [8] in Matlab. Sample matched features for 2 consecutive frames are shown in Figure 3. It should be noted that although some features do not belong to the vehicle, most of them do. In Algorithm 3, we discuss about how to eliminate features that don't belong to the vehicle.

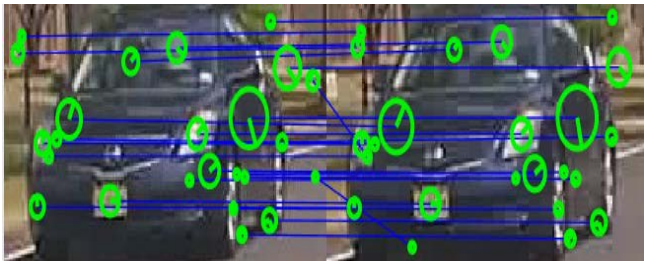


Figure 3: Feature matching between consecutive frames. The images shown above are captured from the bounding boxes.

We also need to note that vehicle count depends on vehicle detection. Vehicles that are not detected for various reasons, are not counted. For example, vehicles that are obscured by other vehicles, could not be counted with our algorithm.

6.3 Lane Estimation

Intelligent vehicle systems enable applications that work with or for the human users with driver-assistance systems. Lane determination is an important concept of these apps and will also be heavily used in autonomous driving. Lane keeping / departure warning systems have already been investigated in the literature and by tracking lane markers, it's possible to determine the lane of travel for the camera vehicle. Our system can compute lane-level traffic estimation while Google Maps/Waze has difficulty due to GPS error (Lane =3.7 meter, but GPS error could be up to 10-20 meters in urban canyons).

By detecting and tracking lane markers, we can identify the traveling lane for the vehicles that travel in the same direction. We propose to create virtual lane regions using the lane markers and determine the traveling lane for each vehicle by intersecting the

vehicle bounding box with the virtual lane regions. These virtual lane regions update with the motion of the car.

However, for vehicles traveling in the opposite direction, we cannot always observe the lane markers due to barriers in between. In order to estimate the lane for vehicles traveling in the opposite direction, we propose to generate pseudo-lane markers that separate the lanes of travel. To generate the pseudo-lane markers, we identify two vehicles from separate lanes, and their corresponding bounding boxes. It is useful to identify these vehicles when one vehicle is partially obstructed by another. The virtual line in between these two bounding boxes is considered to be a pseudo lane marker. By creating these markers, the system identifies lane regions. After creating the virtual lane regions, we repeat the process and determine the traveling lane for vehicles heading in the opposite direction. The lane estimation results will be discussed in Section 7.

6.4 Speed Estimation

For vehicles that are sold in United States since 1996, installation of On-Board Diagnostics (OBD) II[2] port is mandatory to provide self-diagnostics and data reporting capabilities. With this port, in addition to many things, speed information can be accessed. In some cases, this information may be more accurate than GPS based speed estimation, particularly in areas where GPS accuracy is low.

The average speed of vehicles determines the traveling time which is a good measure of road congestion and traffic performance. By knowing the speed of each traveling vehicle, an average speed of that route could be calculated and this information can potentially be used by many applications. One way to achieve this information is by mounting stereo cameras inside cars. These cameras can estimate depth of vehicles in the camera's field of view, at every frame, and hence calculate the vehicles' speed. However, the range of these cameras is usually in the order of 20 meters, and may not be sufficient for vehicles traveling on highways, where distances are larger.

Instead, we propose to use the dashboard mounted camera to estimate the speed of each vehicle. Since we don't know the distance of the car to our camera, we calculate the distance traveled by that vehicle from one frame to another. If we assume the vehicle is traveling parallel to the camera, by using the camera parameters, we can estimate how far the vehicle has moved in real world during that frame. First, we need to calibrate the camera.

Camera Calibration. We obtain the intrinsic camera parameters by using the camera calibration toolbox in MATLAB [5]. With these parameters, we can get the real-world distance to objects in one frame, just by knowing the pixel coordinates of them in the frame. For this reason, we take 20 pictures of a chessboard from different angles and different distances. The other input to the application is the actual size of chessboard square (3 cm). The output of this application is focal length of the camera (both in x and y axis) and intrinsic parameters that are needed to calculate the distance between two objects in the image.

After determining these parameters, one can calculate the relative change in distance for an object in real world, assuming that object moves parallel to the camera by knowing the horizontal distance between the object and the camera. To check the calibration performance, we conducted several tests where multiple objects were placed in a straight line, and a picture is taken by the camera facing parallel to that line. The maximum distance error is less than 2% for both inside and outside experiments.

The next step is to find the speed of the object. After observing an object move about d meters in consecutive frames, the speed of that object is computed as 30×d m/s for a 30 fps camera. For vehicle speed estimation, we can calculate how far each matched feature point moved from one frame to another as described in the Algorithm 3. However, not all the matching features belong to the vehicle and they might move differently. For that reason, we use the distance values that are within one standard deviation of the average distance value for consecutive frames. Since most of the feature points would belong to the car and move similarly, we can eliminate the feature points of non-vehicle objects such as trees, lane markers etc.

One can claim that this speed estimation method works only for the straight roads, and when vehicles travel parallel to the camera vehicle. Although that's a valid argument, even if the road is not straight, vehicles move almost parallel to each other if we consider only a short period of time where two vehicles are close to each other. Therefore, we use last 10 frames of a vehicle in the view of our camera, to calculate the relative distance change. If one vehicle is tracked for more than 10 frames, we only use last 10 frames to estimate how far that vehicle is moved. As a final step, since we know how much time it took for the object to move that distance (by using the specifications of the camera), we can calculate the speed of the object. In Section 7, we present the speed estimation results for three different routes we use.

7 PERFORMANCE EVALUATION

For evaluating our system, we aim to answer the following questions:

- What is the vehicle detection accuracy on roads?
- How does this accuracy change with varying traffic conditions?
- Is it possible to detect lanes of the traveling vehicles?
- What is the speed estimation accuracy?
- How does this system perform compared to GPS-based traffic estimation systems?

To answer these questions, we conducted the following experiments.

7.1 Experimental Setup

Hardware. In the proposed platform, we need a front facing camera that can work with different lighting conditions. After several real-world experiments, we observed that Go Pro camera performance deteriorates with exposure to direct sunlight. However, OmniVision camera gives similar performance results even with the direct exposure of sunlight. The second component is a graphical processing unit (GPU) to process the recorded stream in real-time. With a TX1 board, more than 30 fps could be achieved. The vehicle detection is achieved by a real-time object detection method. We

[2] www.obdii.com

use YOLO: Real-Time Object Detection method to detect vehicles. We conduct our experiments on two different routes in the United States. Route 1 is a campus road with two lanes, one in each direction. Route 2 is a highway in New Jersey with 3 lanes in each direction. In both roads, we use 1 Omnivision OV 10635 camera that is placed on the windshield of a car. We use a server with a GPU, Quadro K-5000 to process the collected images.

Real Road Scenarios. Route 1 is a 0.4 mile stretch of a campus road, depicted in Figure 4(a). We use two vehicles traveling in opposite directions. We placed the camera in one vehicle and use GPS in both vehicles to calculate speed for each vehicle. Since there is only one lane in each direction, we focus on counting vehicles, and estimating the speed of each. We watch collected videos manually to obtain the ground truth for vehicle count, and use GPS-based speed values to get the speed estimation accuracy for our system.

Route 2 is a national highway through New Jersey, shown in Figure 4(b). We covered a distance of 1 mile on this highway during our experiments. We use a total of 4 cars; three of them traveling in the same direction, one after the other, and the fourth car with the camera traveling in the opposite direction. In all cars, GPS is enabled to calculate the speed as ground truth. We focus on counting vehicles in the opposite lane correctly and estimating the speeds of target vehicles with minimal error. We use the speed of known vehicles as ground truth for evaluation. Since there are multiple lanes, we also investigate the lane of travel for each traveling vehicles. For vehicle counting and lane estimation accuracy, we manually label the ground truth.

Metrics. In this work, first, we evaluate the performance of vehicle detection using the following two metrics: (1) detection rate (DR) is the percentage of vehicles that are detected by our algorithm. For example, if x cars are traveling and if our algorithm can detect y of them, then the detection rate becomes (y/x). (2) False detection rate (FD) is the case when our algorithm detects a non-vehicle object as a vehicle.

Second, we evaluate the performance of our lane estimation technique by looking at the detection accuracy. For example, if a target car is traveling in the left most lane and if the algorithm detects the lane of travel as the left, the accuracy is 1, otherwise 0. We evaluate lane estimation for vehicles traveling in both directions.

Third, we evaluate the performance of the speed estimation by calculating the error; the difference between the estimated and true speed of the vehicle. True speed of the vehicle is collected using GPS traces.

7.2 Experimental Results

Counting Accuracy. For Route 1, we drove our vehicle with the camera and detected all vehicles traveling in the opposite direction. Since there is only one lane in the opposite direction, all vehicles were detected and counted perfectly. We performed 10 driving tests on this route, and counted a total of 25 vehicles traveling in the opposite direction.

For Route 2, we drove our vehicle with the camera in the left most lane of the highway in order to observe more vehicles driving on the opposite side. With a total of 4 trips, we observed that all vehicles traveling in the left most lane could be detected, but some vehicles traveling in the middle and right most lane could not be detected because those cars are partially or fully obstructed by another vehicle. Table 2 shows the detection rate and false positive rate for 4 different driving experiments made on that highway. In this table, LL-DR represents the left lane detection rate, ML-DR represents the middle lane detection rate, RL-DR represents the right lane detection rate and FPR represents false positive rate. When the traffic is heavy, our detection rate suffers slightly, since more vehicles in the far away lane are now obstructed by vehicles in the left lane. We define any non-vehicle detection as a false positive. These are generated from the vehicle detection module. However, as mentioned in Section 6.2, we only consider a detection as a vehicle if it is detected in consecutive frames. Since false detections only appear intermittently, they are not counted as vehicles, since they are not detected in consecutive frames.

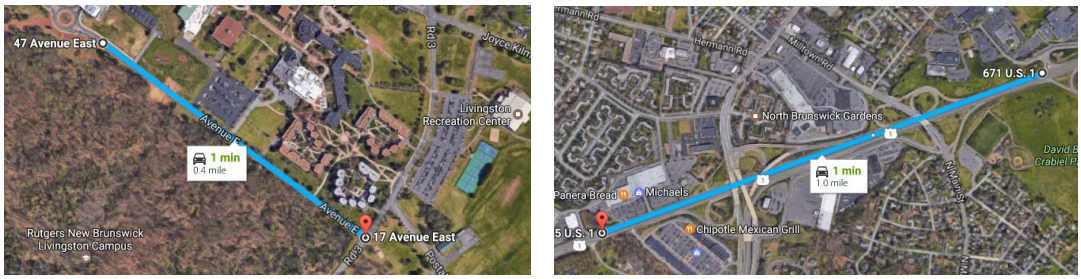
Traffic Condition	LL-DR	ML-DR	RL-DR	Overall-DR	FPR
Light	94.7%	91.6%	90.4%	92.1%	2
Light	100%	88.4%	84%	91.5%	1
Heavy	95.5%	70.8%	62.8%	75%	0
Heavy	98.1%	76.9%	56.3%	78.1%	0

Table 2: Detection performance for Route 2. LL-DR: Left Lane Detection Rate; ML-DR: Middle Lane Detection Rate; RL-DR: Right Lane Detection Rate; FPR: False Positive Rate.

Lane Estimation. To facilitate lane estimation, the system needs to first define the lane regions. To that end, we focus on detecting lane markers in the view of the camera. Figure 5 shows the detected left lane marker with a yellow line and the right lane marker with the purple line. The region in between is the traveling lane used by the camera vehicle. By using the features of the detected lines (i.e., the yellow solid line, white solid line or white dashed line), we can also determine if the traveling lane is the left most lane, one of the middle lanes or the right most lane for multi-lane roads. Using lane markers on the same side of the road as the camera vehicle, we can generate lane regions. For each detected vehicle, the system determines the traveling lane by checking the overlap region of the vehicle's bounding box with the nearest lane region. The lane region with the maximum overlap is said to be the traveling lane for that vehicle.

Unfortunately, for the opposite direction, we cannot always observe the lane markers due to barriers in between. In order to estimate the lane in which the vehicles are traveling in the opposite direction, we propose to generate pseudo-lane markers that separate the lanes of travel. As discussed in Algorithm 2 in Section 5, we create artificial lane markers and then classify the lanes using those markers. Figure 6 shows the number of vehicles per lane calculated by this algorithm for a sample road. Using our algorithm, we can estimate how many vehicles are traveling in each lane, in real time.

For ground truth, we manually count the number of vehicles for each lane and create a confusion matrix to evaluate the performance of this technique for Route 2, as shown in Figure 7. From this figure, we can observe that the estimated lane performance is highest for vehicles that are traveling in the left most lane. For the middle lane, we still have high estimation rate, however, almost one-third of the



(a) Route 1: Campus Road in New Jersey. One lane in each direction.

(b) Route 2: Highway in New Jersey. Three lanes in each direction.

Figure 4: Experiment roads.

Figure 5: The lane of travel.

Figure 6: Vehicle count for each lane. RL:Right Lane; ML:Middle Lane; LL:Left Lane.

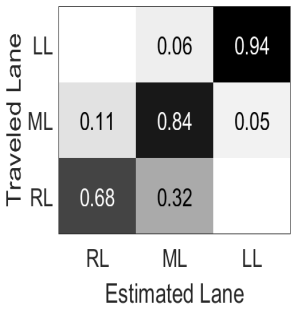


Figure 7: Confusion matrix for Route 2

Relative Speed	Estimated Speed	Error
50	50.5	1.01%
60	61.6	2.71%
70	73.4	4.85%
80	81.4	1.76%
90	93.1	3.4%
95	99.4	4.6%

Table 3: Speed estimation error for Route 1

right lane traveling vehicles are identified as they are traveling in the middle lane.

We want to mention that lane estimation needs a manual guidance that needs to be done once, for each road. In this process, two random vehicles need to be tracked from each lane and then pseudo-lane markers are created by using the trajectories of these vehicles. Finally, lane estimation is achieved by comparing the center coordinates of each new vehicle with the lane markers.

Speed Estimation. As we discussed in Section 6.4, by using the intrinsic camera parameters and the horizontal distance between the camera and the target car, we can calculate the speed of the target car.

In Route 1, there is only one lane in each direction, therefore, the horizontal distance between the target car and the vehicle is 3.7 m, which is the average lane width in the United States. Both the target car and the camera car are driven at different speeds and encountered 6 times in Route 1. Table 3 shows the speed estimation results for this road. The first column represents the relative speed of two vehicles compared to each other. For example, if the target car is traveling at 25 mph and the camera car is traveling at 35 mph, the relative speed becomes 60 mph. Table 3 shows that the speed estimation works well for a variety of relative speeds and is always less than 5%.

Since Route 2 is a major highway, we could drive with higher speeds and extend the results for speed estimation that we obtained from Route 1. In this test, all target cars are traveling in the left most lane of the opposite side at different speeds. We performed the test with 3 vehicles driving in the opposite direction. The

Relative Speed	Estimated Speed	Error
100	108.3	8.32%
110	116.7	6.01%
120	125.5	4.58%
130	139.2	7.07%
140	152	8.57%

Table 4: Speed estimation error for Route 2

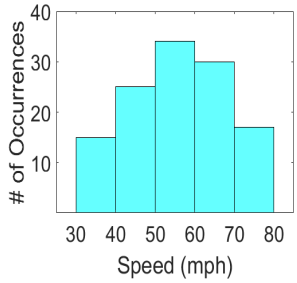


Figure 8: Speed values for vehicles on Route 2

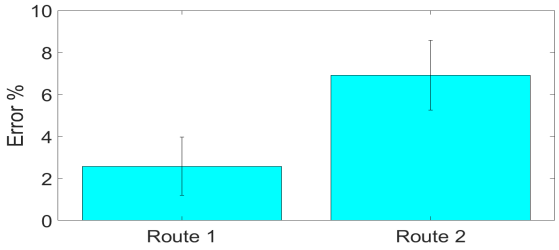


Figure 9: Speed estimation error for all roads.

results are presented in Table 4. For Route 2, the error in the speed estimation is generally larger than for Route 1, but still less than 9%. One possible explanation could be that there is a large barrier separating opposite traffic in Route 2, that blocks the lower parts of the vehicles and causes fewer feature points to be detected, and be used for speed estimation. Also, the distance between target car and the camera car is also higher (6 m.) compared to Route 1 (3.7 m.).

In Figure 8, we show the speed values used by drivers for two highways. In Route 2, vehicles are driven between 30-80 mph. For majority of the drives, speed values of 50-60 mph have been used. These results also prove that using GPS-based speed calculations for a few drivers could give very different results, so GPS-based traffic estimation may not be accurate.

In Figure 9, we show the error obtained in speed estimation for all three roads. For Route 1, which is a one-lane road in each direction, we have about 2.5% error in the speed estimation. For Route 2, the error is slightly higher, 6.9%. This is because, in Route 2 traffic for each direction is separated by a big barrier, and therefore some parts of the vehicles in the opposite direction are obscured. It affects our speed estimation because we are using feature matching between consecutive frames, and more matched feature points enable better estimation.

8 DISCUSSION

We have presented the design, implementation, and evaluation of a traffic estimation technique using live video streams collected from a moving vehicle. Unlike traffic surveillance camera based works, our work aims at detecting vehicles using front facing cameras in vehicles in real-time. When driving a vehicle in the left most lane, we have a clear view of the cars driving in both directions and it enables us to count the maximum number of cars on roads.

In estimating the speed of vehicles, we employ camera parameters and calculate the relative distance change of vehicles between frames assuming that vehicle is moving parallel to the camera. One might claim that roads are not always straight so this assumption might not be valid. However, we only focus on 10 frames before the vehicle leaves the camera view, and for this duration, the movement of the vehicle is almost parallel to the camera.

One might argue that the hardware needed for our system could limit the total number of users. Deploying such a system to public buses, taxis or patrol vehicles could extend the usage of the system widely.

Our system could easily be used by enforcement forces. Since the speed estimation works with the error less than 12% even for the roads with wide barriers, speed monitoring could be managed when patrolling on the road. With this way, wide coverage could be achieved at a low cost compared to using speed radars or speed guns. Our system could also be extended to provide a platform for environment-to-car communication, based on camera view [9].

It is easy to think that GPS-based navigation systems can also calculate the speed, so why such a vision-based system would be needed. With those systems, speed is calculated by using a mobile device traveling on road with GPS activated. Due to battery problems, a lot of users do not activate GPS for a long period of time. Even if they do, those navigation systems can only calculate the speed experienced by that mobile phone which could easily give an outlier speed. For example, if the GPS is enabled for a motorcycle driver, the estimated speed would most likely be higher than the average speed of vehicles, or if the GPS is enabled for an old driver, the estimated speed would be lower than the average speed of vehicles. Also, GPS accuracy suffers in urban environments [25]. On the other hand, by using our system, we can estimate the speed of each vehicle. Therefore we can calculate the average speed of vehicles on that road.

Depending on the application scenario, having a detection accuracy less than 100% may not be enough. Since we are using front facing cameras that are deployed in the vehicle, some vehicles traveling in the opposite direction, especially traveling in the right most lane, could be obstructed by other vehicles in between. Since the vehicle is partially observed by the camera, our object detection component may not identify the object as the vehicle. However, all vehicles that are traveling in the left most lane of the other direction would be identified as long as the barrier between the lanes is not high. By counting the number of vehicles traveling in the left most lane of the other direction, we can still determine the traffic

conditions. Also, by using a camera that is placed on the roof of the vehicle, the vehicle count accuracy could be increased.

9 CONCLUSION

In this work, we use vehicles as edge compute nodes and estimate the traffic from video streams using front facing cameras. With a real-time deep neural network object detection method, we can detect vehicles in both directions and count the number of vehicles traveling. This number could give much meaningful information about the traffic when combined with the estimated speeds of vehicles. Such information could be shared with neighboring vehicles or a map service. A service that collects such information from large numbers of vehicles could create an up-to-date map with fine-grained, near-real time vehicle positions for that road. This results in more accurate and sophisticated assessment of roads, compared to traffic surveillance cameras, that do not cover all roads. With such a system, traffic flow can be enhanced and route planning for new cities could be achieved.

We used a dash-camera to collect the footage of the traffic and a GPU-equipped laptop for real-time vehicle detection and tracking. Specifically, we have shown that all oncoming vehicles could be detected, counted, and tracked with less than 5% speed estimation error in two-lane campus road trials. In highway experiments, we achieved a minimum 75% vehicle counting accuracy under crowded traffic conditions, and over 90% accuracy for light traffic conditions. The speed estimation error is about 9% for the highway. This error can further be decreased for most common traffic roads, with shorter median strips. For example, we observed about 3% error in speed estimation for campus road experiments with no median strip on the road. We also have shown that the lane of travel for each vehicle could be estimated in both directions. Specifically, we could estimate the lane of travel perfectly for the vehicles that are traveling in the same direction with the camera vehicle, and over 90% for the vehicles traveling in the left most lane of the opposite direction. Future work can integrate these techniques into a live traffic-view of real-time vehicle locations and speeds.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No CNS-1329939.

REFERENCES

[1] Analyses of rear-end crashes and near-crashes. https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/analyses20of20rear-end20crashes20and20near-crashes20dot20hs2081020846.pdf. Accessed: 2017-03-18.
[2] CUDA quadro-k5000. http://www.nvidia.com/object/quadro-k5000.html. Accessed: 2017-03-17.
[3] Hough transform. http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm. Accessed: 2017-04-09.
[4] Nvidia drive px. http://www.nvidia.com/object/drive-px.html. Accessed: 2017-04-23.
[5] Single camera calibration app. https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html. Accessed: 2017-03-18.
[6] Stanford university cars dataset. http://ai.stanford.edu/~jkrause/cars/car_dataset.html. Accessed: 2017-03-17.
[7] Uiuc image database for car detection. https://cogcomp.cs.illinois.edu/Data/Car/. Accessed: 2017-03-17.
[8] Vlfeat. http://www.vlfeat.org/index.html. Accessed: 2017-03-18.
[9] A. Ashok, S. Jain, M. Gruteser, N. Mandayam, W. Yuan, and K. Dana. Capacity of pervasive camera based communication under perspective distortions. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 112–120, March 2014.
[10] Erhan Bas, A Murat Tekalp, and F Sibel Salman. Automatic vehicle counting from video for traffic flow analysis. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 392–397. Ieee, 2007.
[11] Massimo Bertozzi, Alberto Broggi, Alessandra Fascioli, Thorsten Graf, and M-M Meinecke. Pedestrian detection for driver assistance using multiresolution infrared vision. *IEEE transactions on vehicular technology*, 53(6):1666–1678, 2004.
[12] David Beymer, Philip McLauchlan, Benjamin Coifman, and Jitendra Malik. A real-time computer vision system for measuring traffic parameters. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 495–501. IEEE, 1997.
[13] Lorenzo Bruzzone, Diego F Prieto, and Sebastiano B Serpico. A neural-statistical approach to multitemporal and multisource remote-sensing image classification. *IEEE Transactions on Geoscience and remote Sensing*, 37(3):1350–1359, 1999.
[14] Daniel J Dailey, Fritz W Cathey, and Suree Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):98–107, 2000.
[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.
[16] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2012.
[17] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pages 647–655, 2014.
[18] Jacob Velling Dueholm, Miklas Strøm Kristoffersen, Ravi Kumar Satzoda, Thomas Baltzer Moeslund, and Mohan Manubhai Trivedi. Trajectories and maneuvers of surrounding vehicles with panoramic camera arrays. *IEEE Transactions on Intelligent Vehicles*, 1(2):203–214, 2016.
[19] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
[20] Dariu Gavrila. Pedestrian detection from a moving vehicle. *Computer VisionâĂŤECCV 2000*, pages 37–49, 2000.
[21] Dariu M Gavrila and Stefan Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International journal of computer vision*, 73(1):41–59, 2007.
[22] Wolfgang S Homburger and James H Kell. Fundamentals in traffic engineering. 1988.
[23] W-L Hsu, H-YM Liao, B-S Jeng, and K-C Fan. Real-time traffic parameter extraction using entropy. *IEE Proceedings-Vision, Image and Signal Processing*, 151(3):194–202, 2004.
[24] Albert S Huang. Lane estimation for autonomous vehicles using vision and lidar. 2010.
[25] Shubham Jain, Carlo Borgiattino, Yanzhi Ren, Marco Gruteser, and Yingying Chen. On the limits of positioning-based pedestrian risk awareness. In *Proceedings of the 2014 Workshop on Mobile Augmented Reality and Robotic Technology-based Systems*, MARS '14, pages 23–28, New York, NY, USA, 2014. ACM.
[26] Shubham Jain, Viet Nguyen, Marco Gruteser, and Paramvir Bahl. Panoptes: Servicing multiple applications simultaneously using steerable cameras. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 119–130, April 2017.
[27] Young-Kee Jung and Yo-Sung Ho. Traffic parameter extraction using video-based vehicle tracking. In *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, pages 764–769. IEEE, 1999.
[28] SamYong Kim, Se-Young Oh, JeongKwan Kang, YoungWoo Ryu, Kwangsoo Kim, Sang-Cheol Park, and KyongHa Park. Front and rear vehicle detection and tracking in the day and night times using vision and sonar sensor fusion. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2173–2178. IEEE, 2005.
[29] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
[30] Joel C McCall and Mohan M Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE transactions on intelligent transportation systems*, 7(1):20–37, 2006.
[31] Takashi Okano. Method for correcting pixel data in a self-luminous display panel driving system, February 15 2000. US Patent 6,025,818.
[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
[33] Todd N Schoepflin and Daniel J Dailey. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):90–98, 2003.
[34] Chris Urmson et al. Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23(2), 2008.

[35] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613. IEEE, 2009.

[36] Paul Viola and Michael J Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[37] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4705–4713, 2015.

[38] Zhigang Zhu, Bo Yang, Guangyou Xu, and Dingji Shi. A real-time vision system for automatic traffic monitoring based on 2d spatio-temporal images. In *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*, pages 162–167. IEEE, 1996.