CoDrive: Improving Automobile Positioning via Collaborative Driving

Soteris Demetriou
University of Illinois at Urbana-Champaign
sdemetr2@illinois.edu

Puneet Jain
Hewlett-Packard Labs
puneet.jain@hpe.com

Kyu-Han Kim
Hewlett-Packard Labs
kyu-han.kim@hpe.com

***Abstract*—An increasing number of depth sensors and surrounding-aware cameras are being installed in the new generation of cars. For example, Tesla Motors uses a forward radar, a front-facing camera, and multiple ultrasonic sensors to enable its Autopilot feature. Meanwhile, older or legacy cars are expected to be around in volumes, for at least the next 10 to 15 years. Legacy car drivers rely on traditional GPS for navigation services, whose accuracy varies 5 to 10 meters in a clear line-of-sight and degrades up to 30 meters in a downtown environment. At the same time, a sensor-rich car achieves better accuracy due to high-end sensing capabilities. To bridge this gap, we propose CoDrive, a system to provide a sensor-rich car's accuracy to a legacy car. We achieve this by correcting GPS errors of a legacy car on an opportunistic encounter with a sensor-rich car. CoDrive uses smartphone GPS of all participating cars, RGB-D sensors of sensor-rich cars, and road boundaries of a traffic scene to generate optimization constraints. Our algorithm collectively reduces GPS errors, resulting in accurate reconstruction of a traffic scene's aerial view. CoDrive does not require stationary landmarks or 3D maps. We empirically evaluate CoDrive which is shown to achieve a 90% and a 30% reduction in cumulative GPS error for legacy and sensor-rich cars respectively, while preserving the shape of the traffic.**

I. Introduction

Global positioning system (GPS) has revolutionized the way people navigate. Nonetheless, despite an abundance of GPS devices at our fingertips, navigation in urban canyons and under heavy traffic remains challenging. Urban canyons are challenging because GPS positioning is highly inaccurate. Due to multipath interference, GPS accuracy in urban canyons ranges between 30-50 meters [1], [2]. Similarly, heavy traffic is challenging because current devices lack lane awareness. The accuracy of 5-10 meters in a clear line-of-sight is insufficient for lane-level guidance [3]. As a result, these conditions put an additional cognitive burden on drivers, which often results in wastage of time, fuel, and money, and sometimes accidents and deaths [4], [5], [6]. The goal of this paper is to address these problems, more-so in a futuristic scenario.

Recent developments suggest that the automobile industry is taking a bold leap into connected vehicles and autonomous driving. In fact, the growth of next-generation-car sales is expected to be rapid, with an estimated 45% of cars on the road by 2020 to be connected [7]. Moreover, various levels of autonomy have been already announced, tested, or launched. Such cars feature advanced sensing capabilities, including multiples of range sensors (Lidar and Radar), 360° cameras [8], onboard GPUs [9], and high-speed connectivity [10], [11]. To enable seamless driving, these hardware capture and process gigabytes of sensor data in real-time.

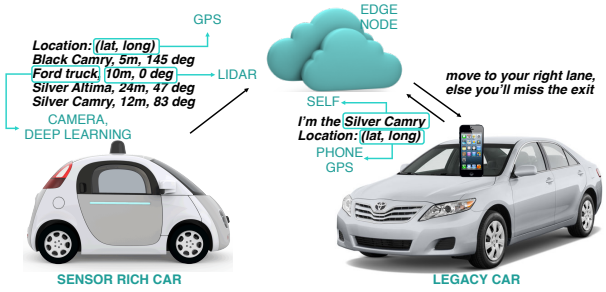


Fig. 1: CoDrive example scenario

This work leverages the increasing presence of such hardware in a rather contrary way, i.e., instead of helping a sensor-equipped car in navigating better, we use it to improve positioning of a legacy car. This intuition stems from a well-established technique called simultaneous localization and mapping (SLAM), where a robot resets its measurement error using static landmarks and dead reckons between them using on-board sensors. In our case, past works have shown that it is possible to track vehicle dynamics (lane change and turns) using inertial sensors and GPS of the smartphone [12]. However, previous approaches make several impractical assumptions which hinder their universal applicability.

Specifically, previous approaches assume that the start location and lane position of the vehicle are known. Moreover they assume that the vehicle always enters a road segment from the right-most lane. In addition, sensor-based approaches, ignore accumulation of long term dead-reckoning errors and drift [2], [12]. Other previous works use a front-facing camera to mitigate this issue; however, they require the camera-bearing apparatus (e.g. a smartphone) to be mounted in a specific manner [13], [14]. While both approaches are promising, for them to work universally, a reliable mechanism is required to periodically reset their estimation errors. By building CoDrive, we enable such a mechanism. Similar to previous works, CoDrive leverages GPS and inertial sensors of smartphones in legacy cars for continuous tracking. In contrast with them, CoDrive does not make any assumptions regarding a vehicle's initial positioning and/or orientation.

CoDrive achieves precise positioning whenever it encounters a sensor-rich car. Each subsequent encounter is then

used to reset dead-reckoning errors and improve positioning estimates. Figure 1 exemplifies CoDrive: whenever a sensor-rich car detects a vehicle, it shares its own GPS coordinates, the angle and distance of the observation and the visual fingerprint of the detected vehicle with the nearest compute node. The compute node leverages these measurements along with the detected vehicle's own estimation of location (e.g. GPS measurement from the driver's smartphone) to correct the detected vehicle's location. In this way, CoDrive leverages the increasing presence of sensor-rich cars in urban environments to eliminate the need for and the cost of deploying and managing static landmarks for positioning corrections.

An interesting aspect of CoDrive is that it performs better under dense environments than sparse. In dense traffic, higher navigation accuracy is needed because lane switching becomes challenging. A system which could assist in correct lane keeping and generate advance alerts is highly desirable. Similarly, in urban canyons (e.g. a downtown situation), traditional GPS performs poorly due to high multipath errors (see Figure 2 for an example of the effect of GPS errors in downtown San Francisco) and a mechanism is needed to reset them. CoDrive offers both these features but it requires participation of at least one sensor-rich car. It is not surprising that CoDrive performs better with more participation. During early adoption—like any crowd-sourcing approach—we believe this is indeed a limitation, especially given that it does not offer consistent accuracy guarantees. However, one should note that CoDrive offers benefits where they are needed the most. Navigation in sparse environments is easier, therefore accuracy beyond traditional GPS does not hold substantial merit. On the other hand, navigation in urban environments is challenging and all enabling factors for CoDrive exist there. CoDrive is an ideal fit for urban environments where there is higher accuracy demand, traffic density, and more early technology adopters.

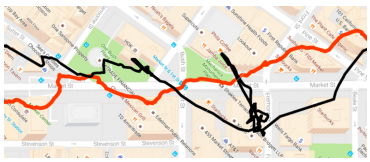


Fig. 2: GPS errors in San Francisco downtown while walking on a straight line on Market St (2 trials).

Moreover, beyond precise positioning, CoDrive builds a means of communication between the two different generations of cars. In CoDrive, vehicles communicate their surrounding information to the closest edge computing node. The edge node then performs matching over the received visual, sensory, and coarse location cues. CoDrive's main contribution stems from its novel optimization framework, which combines such information across different generations of cars to generate improved positioning for each of them. Given that CoDrive enables inter-car communication without new infrastructure, it can be further utilized to convey other information related to accidents, weather, or road conditions, to the trailing vehicles in traffic. We believe this synergy between the two generations is of great value with conspicuous benefits in the years to come. Note that the average age of a vehicle in the US is 11.5 years [15]. Therefore, in absence of significant improvement cost, millions will lack advanced features for at least a decade.

Our system CoDrive (collaborative driving) enables collaboration among cars and facilitates an open-car ecosystem. This ecosystem is different from the existing, where a sensor-rich car uses its hardware to improve its own location accuracy. Naturally, for such an ecosystem to succeed, the benefits of collaboration should be mutual. While benefits are apparent for legacy car owners, the same might not be true for sensor-rich car owners. CoDrive produces value for both kinds: in a downtown scenario, we find CoDrive reduces the location errors of legacy cars from an average of 49 meters to 4 meters, while also improving the accuracy of sensor-rich cars by 3x. In case this incentive is proven inadequate for sensor-rich car owners, the service-provider could turn to an Uber-like business model, where sensor-rich car owners benefit financially for sharing their sensor information. Given that manufacturers would soon have fleets of sensor-rich cars on the roads [16], [11], [8], [10], they could collect measurements and utilize CoDrive to bootstrap a variety of driving assistance applications. Finally, CoDrive can reconstruct the aerial view of the traffic and enable fine-grained travel behavior.

Our contributions. We summarize our contributions below:

- The first system which uses the existing sensing capabilities of cars as moving landmarks for correcting positioning errors.
- A novel optimization framework and inter-car communication technique for improving outdoor positioning of all participating cars, combining distance, angle, and visual information in a unique way.
- A comprehensive system architecture, end-to-end implementation, and real-world evaluation with commodity hardware.

Paper Organization: In Section II, we provide an overview and describe individual components of the system. Section III presents our optimization technique for precise positioning of cars and Section IV the evaluation of CoDrive.Section V covers the related work and Section VI concludes the paper.

II. System Design

Approach: Our goal is to design an end-to-end system for correcting GPS errors in urban environments. One could argue that differential GPS (DGPS) could be appropriate in this case. However, while DGPS can be accurate up to 10 cm, it has two major disadvantages. Firstly, it performs poorly in urban canyon environments: [2] found DGPS's average error to be 75m which can degrade up to 200m. Secondly, it requires a custom receiver to be installed on legacy cars. Alternatively, vision-based methods can be used to reduce GPS errors, for example by leveraging lane markers [13] or road-signs [14], while others have used mobile sensing methods to detect relative positioning, route turns, potholes, and stop-signs [17], [18], [2]. Unfortunately, vision-based methods require a phone

to be mounted in a specific way to ensure clear lane-view – which is impractical. Moreover, a phone can only identify lane markings of the closest lanes due to its narrow field-of-view (typically < 54 degrees); identifying the lane position requires a clear view of the entire road, which is not possible in a dense traffic environment. On the other hand, sensing-based methods do not scale because of (a) noise in inertial sensors, and (b) significant pre-deployment effort (environment wardriving requirements for identifying repeatable landmarks (e.g., potholes), lanel-level digital maps, auxiliary equipment). In our work we propose a solution which is appropriate for urban environments, uses no special infrastructure in legacy cars other than a driver's smartphone, makes no assumptions regarding the placement of the smartphone in a vehicle, and does not require laborious and costly pre-deployment effort.

Our system, CoDrive, is an outdoor vehicle positioning system which leverages combined sensing capabilities of sensor-rich and legacy cars to correct estimation errors. In particular, on an opportunistic encounter, measurements (color, depth, and location) from the sensor-rich car(s) and GPS location of the legacy car(s) are streamed to the CoDrive edge computing node. The edge node uses a novel location optimization framework (Section III) which transforms raw observations into constraints of an optimization problem. A solution of this problem results in accurate positioning of all participating cars. The corrected locations are streamed back to the participating cars which use it to reset their positioning errors. CoDrive's system architecture comprises of three modules: a cloudlet residing in sensor-rich cars, an edge node at cell tower base stations, and a smartphone inside legacy cars. Next, we discuss different trade-offs and the rationale behind our design decisions.

System Design Rationale: Various metrics such as accuracy, latency, and bandwidth play an important role when designing a networked system. These metrics influence the location of computation: the cloudlet, the edge, or the cloud. In our case, a sensor-rich car can generate gigabytes of visual data every second. However, due to limited *upload bandwidth*, offloading entire data to the edge or the cloud is not possible. Therefore, CoDrive processes visual data on the cloudlet and only offloads the inferences. CoDrive location optimization requires inferences from all participating cars to be co-located. Therefore, the computation can be performed either at the edge or the cloud. CoDrive uses the edge due to smaller *round-trip latency*and lower *monetary cost*. The round-trip latency between a car and the edge is typically small compared to the cloud. This is because of geographic proximity of the edge (single-hop) over the cloud (multi-hop). More benefits of edge computing over the cloud are documented here [19]. We also decided not to focus on vehicle-to-vehicle communication (V2V) due to privacy challenges, long connection setup times, fleeting opportunity between vehicle encounters, and because all vehicles will be required to integrate special equipment.

Figure 3 shows various components of our system. The cloudlet acquires data from various sensors and processes

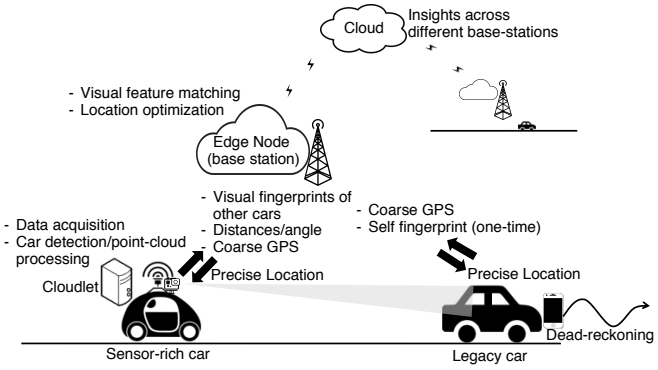


Fig. 3: CoDrive Architecture

them locally. The processing includes car detection and visual fingerprinting; overlaying depth pointcloud on RGB and; distance/angle estimation for each car. The visual fingerprints, distance/angle, and GPS reading are then uploaded to the edge node. The edge node first correlates visual fingerprints with all participating cars in the area. Then it runs the location optimization resulting in precise positioning for both the sensor-rich and the legacy cars. The cars then reset their positioning based on the newly received information and locally track vehicle dynamics such as turns and lane changes—on legacy cars this is performed on the driver's smartphone.

One of the primary challenges in our work is getting access to a programmable sensor-rich car. Since this was infeasible, we imitate one by mounting sensor hardware on a legacy car. We experiment with various mount designs, several cameras, an array of circularly mounted IR/acoustics range finders, low-cost servo motor based spinning range-finders etc. However, the lack of real-time streaming from cameras, inaccuracies in IR/acoustics distance measurements, and wind friction led us to our converged setup based on circularly mounted phones and a Velodyne lidar at their center (Figure 4).

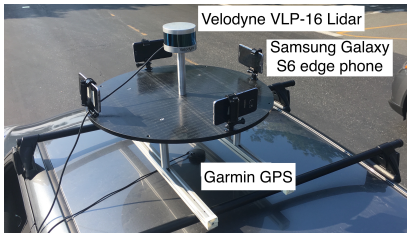


Fig. 4: Sensor-rich car used in our experiments

Computation in the Cloudlet: CoDrive leverages the following information from a sensor-rich car: (a) 3D pointcloud, (b) 360° images, and (c) GPS locations. In the next subsections, we elaborate on the components of the CoDrive cloudlet pipeline and how they are used to infer a better location.

A sensor-rich car should be able to detect cars in a road scene. With our measurement kit mounted on a legacy car (mimicking sensor-rich car behavior), we experiment around our institution. For simplicity, acquired sensor data is stored in the cloudlet and processed later. Our pipeline to process acquired data consists of several image and point-cloud

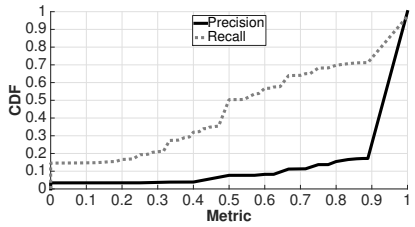


Fig. 5: Car detection accuracy on the *kitti* dataset

processing modules. We first extract cars from the stream of video frames – equivalent to the object-localization in computer vision. CoDrive requires extremely high accuracy and robustness across different view-points. To this end, our natural choice is to use neural networks. Neural networks (a.k.a. deep learning) have made tremendous progress in recent years – outperforming most feature-based detection methods. We use Faster-RCNN [20] implemented on CAFFE [21] to identify cars in a scene. However to improve the object-localization speed and achieve real-time performance, we make two enhancements in the object-localization pipeline. (a) Instead of performing object-localization on every frame, we use object-tracking between every nth frame [22] . We do this because object-localization per frame is slow and takes about $250 - 300$ ms, while the video streams are typically recorded at 30 frames per second. (b) We train a custom deep neural network on the Stanford car dataset [23]; this network is specifically designed to detect cars at higher accuracy and speed across diverse view-points. Our single-class custom trained object-localizer performs substantially better in speed and robustness over the out-of-the-box pre-trained models.

Figure 5 shows the accuracy of car detection on the *kitti* dataset [24]. This dataset is different from the one used in training and it contains scenes with multiple cars in them. The dataset contains a total of 7481 images. Each data point in the graph represents a precision/recall value for an image. We define precision as the fraction of correctly identified cars over the total identified cars. Not surprisingly, deep-learning achieves *near-perfect* precision in the median case. Similarly, recall is the fraction of correctly identified cars over the total number of cars in a scene. We achieve $> 50\%$ recall in the median case. While this may appear unimpressive, it is not a bottleneck in our system's functioning. We inspected poor recall cases closely and found that deep-learning identifies all cars in the direct view correctly. Most unidentified cases are due to heavy occlusions and far distance—could be missed out by lidar's range sensing as well.

For each detected car, we construct a visual fingerprint. A visual fingerprint consists of vision-based features, which remain robust across significant angular changes. Visual fingerprinting is needed for two reasons: (a) to merge all observations of a participating car from different view-points of sensor rich cars, and (b) to map sensor-rich cars' observations with the GPS of the participating car on which the observations were made. We use the HSV spatiogram [25] and car categories (sedan, SUV, etc) [23] to construct a unique visual fingerprint. This fingerprint is used to identify a car from different viewpoints.

After car detection, the next step is to estimate the distance/angle of participating cars with respect to the lidar. To achieve that, we overlay the lidar pointcloud on top of RGB frames. This process is not straightforward because the lidar and RGB cameras are separated by a distance on the mount. Therefore, a transformation between the two needs to be estimated. In computer graphics, such a transformation can be achieved by the combination of two matrices: *rotation* (R) and *translation* (T). We adopt techniques discussed in [26] to estimate these values. Once estimated, any 3D point (X, Y, Z) in the lidar pointcloud can be converted to their respective pixel coordinates (x_p, y_p) by multiplying it with the rotation matrix R and adding translation T to it. For a given configuration, once R and T are estimated, they remain fixed unless the placement (mount in our case) is changed. Due to excessive cost ($>$ \$30K) of 360° cameras, we use multiple phones to emulate 360° awareness; however, the calibration is performed only once. We calculate R and T for the other phones leveraging the mount's angular symmetry. Once a lidar pointcloud is overlaid on RGB frames, we can estimating distance/angle of an individual car. We first sub-select the pointcloud falling under the detection bounding box of a car. Since multiple points fall within this box, the point closest to the center pixel is taken as the representative distance/angle.

The CoDrive cloudlet is implemented in the robotic operating system (ROS). In Figure 6, each box is implemented as a ROS node. Each node publishes and subscribes to one or multiple topics. The data from three sensor nodes (PointCloud, Image, and Location) are received asynchronously at the Master node. We use the `ApproximateTime` algorithm in ROS to nearly-synchronize these streams and to create batches of triplets. These batches are then passed through a computer vision module which outputs visual fingerprints of the cars, distance, and angle. Finally, this information is uploaded to the edge for further processing.

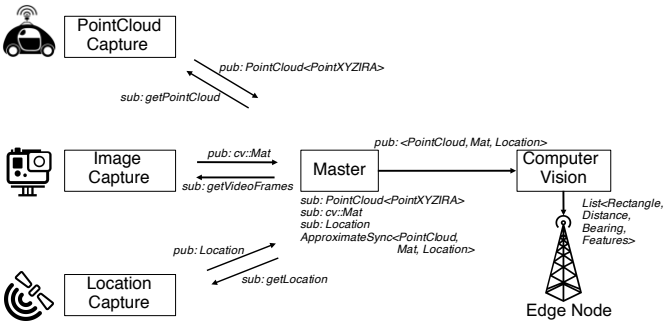


Fig. 6: Processing pipeline of the CoDrive cloudlet

Computation at the Edge: The output of the cloudlet pipeline is uploaded to the edge node. Furthermore, legacy cars use the CoDrive navigation app to report their visual fingerprint during enrollment. In addition, the navigation app reports the cars' GPS locations in real-time. The CoDrive edge node collects this information and performs two main operations: (a) visual

fingerprint matching, and (b) location optimization on the coarse GPS locations.

Visual fingerprint matching is a straightforward process. Each legacy car user registers with CoDrive by specifying their car's make, model, and year. Since fingerprints of legacy cars do not change, they are precomputed and stored at the edge. We automatically map legacy cars' GPS location to a fingerprint when it is streamed to the edge. We correlate visual fingerprints reported by all cars with each other, i.e., those received from sensor-rich cars and those precomputed. Based on the correlation value, observations around a car are joined and forwarded to the next stage for location optimization.

The edge solves a non-linear location optimization problem. Let a *road scene* be a connected graph with cars as nodes. An edge is a distance/bearing measurement from a sensor-rich car (source) to another car (destination). Then, for every road scene, the edge constructs a set of constraints based on sensor-rich cars' observations and plausible roads near the collected GPS measurements and generates an objective function aimed to minimize the positioning error. It then feeds the constraints and the objective function to the location optimizer which—using the *Differential Evolution* method—finds new positions for all cars (Section III).

Computation on the Phone: Our phone side module is lightweight—a navigation app which reports GPS locations to the edge and a service to detect steering maneuvers from inertial sensors. Essentially, after obtaining precise positioning from the edge, our app identifies steering maneuvers such as a lane change or turn to track lane-position changes. We use the gyroscope to detect events such as turns and lane changes [12]. However, unlike [12] which uses peak-detection with hard-coded thresholds, we use a neural network to train a model around each event [27]. We then use a fixed length gyroscope time series data to classify and flag events in real-time.

III. Location Optimization

CoDrive utilizes the advanced capabilities of sensor-rich cars and the coarse location estimates of legacy cars to accurately—within a few meters—recreate a road scene. For example, consider the following scenario: two sensor-rich and two legacy cars are moving on the same road and direction. Both sensor-rich cars detect all other cars in the scene and estimate distances/angles using the techniques presented in Section II. We represent a road scene as a connected graph. When a sensor-rich car, detects another car at a distance and angle from itself, it translates that into an edge. The edge has the sensor-rich car as the source node and the detected car as the destination node. Given this graph, we attempt to answer the key question: *how can we improve the positioning accuracy of all cars on a road?*

In our work, we model traffic as an optimization problem. Intuitively, our goal is to minimize the overall GPS error of participating cars in a road scene. The solution we seek must respect, or is constrained by, the observations of sensor-rich cars and the self-estimated GPS locations. We also aim to preserve the shape of the traffic as much as possible, which is paramount in making lane-level inferences. Additionally, our solution must be independent of any stationary landmarks, road signs, lane markings, or any road conditions.

In particular, CoDrive aims to minimize the cumulative positioning error of all cars that participate in a road scene. Every car in the scene has at least one self-reported location from GPS, in addition to locations reported by sensor-rich cars around it. Our solution minimizes the distance between all reported locations. However, we do that cumulatively since a solution for one car can affect the other. Formally, let $C_{i,j}$ be the location of car i reported by car j, where $C_{i,j}$ comprises of an easting (x) and a northing (y) [1]: $C_{i,j} = \langle c^x_{i,j}, c^y_{i,j} \rangle$. Now, let V_i be the set of *Views* of car i, where the cardinality of V_i, $|V_i|$, equals the number of sensor-rich cars that detected car i. V_i is then a set that consists of *Views* v^j_i of car i by car j. Also, each such *view* consists of the distance between j and i (d^j_i) and the bearing or heading [2] (θ^j_i) of the observation. Thus $v^j_i = \langle d^j_i, \theta^j_i \rangle$.

In a trivial case, where a car reports it's own location and it is not detected by anyone else, the location error would simply be the distance between the true location K_i and the self estimated location $C_{i,i}$: $E_i = \|K_i, C_{i,i}\|$. However, when there are multiple *Views* of a car's location, the cumulative error for car i becomes: $E_i = \|K_i, C_{i,i}\| + \sum_{j \in V_i} \|K_i, C_{i,j}\|$, $i \neq j$, which is equal to the sum of distances from the geometric median of all reported locations. Therefore, our end goal is to find the true locations $K_i, \forall i$, that minimize the following objective function:

$$\text{Objective Function} \quad \sum_{\forall i} (\|K_i, C_{i,i}\| + \sum_{j \in V_i} \|K_i, C_{i,j}\|), i \neq j \tag{1}$$

However, when a sensor-rich car detects another car at a specific distance and angle, its accuracy confidence is inherited in the estimation of the detected car's position. The detected car also reports its own location with a different confidence. Therefore, the true location of each detected car lies within multiple overlapping confidence regions. Formally, we first restrict K_i to be within the reported accuracy range of the GPS of car i. We then take each reported location $C_{i,j}$ as the center of a circle whose radius e_j is defined by the location confidence of car j. Now, K_i needs to be at the intersection (or within all) $C_{i,j}$-centered circles. We include these conditions in the *GPS Constraints*:

$$\text{GPS Constraints} \quad \forall i, j : \|C_{i,j}, K_i\| - e_i \leq 0 \tag{2}$$

As shown in Section II, sensor-rich cars can locate other cars and report the distance and the angle between themselves and the observed car. We translate distance and angle into a set of constraints for our optimization. Formally, these constraints

[1] The coordinates are transformed to UTM.

[2] Bearing indicates the clockwise angle in degrees from North.

dictate where a point $\langle k_i^x, k_i^y \rangle$ can be placed, given the $C_{i,j}$ and v_i^j. We define the *Distance and Angle Constraints* as:

$$\begin{aligned} &\text{Distance and Angle Constraints} \quad \forall i : \forall v_i^j, k_j^x + (d_i^j * \cos \phi_i^j) = k_i^x, \\ &\forall i : \forall v_i^j, k_j^y + (d_i^j * \sin \phi_i^j) = k_i^y \end{aligned} \tag{3}$$

Where, ϕ_i^j is the angle of the v_i^j *view* in anti-clockwise direction from UTM eastings. Thus, given the heading θ^r in radians one can calculate ϕ^r as follows: $\phi^r = ((\frac{\pi}{2} - \theta^r) + 2\pi) \mod 2\pi$. However, it might be the case that some cars have multiple *views*. The distance and angle constraints are then relaxed by allowing the solution to be within a rectangle, spanning from the minimum to the maximum observed values.

Due to the previous constraints, a plausible solution should retain the shape of the scene but could end up on a wrong street or on a building. We improve further by constraining the solution within road boundaries. We find the road closest to the majority of the reported locations and generate a new set of constraints called *Road Constraints*. Formally, let $V = \langle v_1, v_2, ..., v_n \rangle$ be a set of location points and $E = \langle e_{v_1,v_2}, e_{v_2,v_3}, ..., e_{v_n,v_1} \rangle$ be a set of lines connecting location vertices. A road region is then described by a polygon $\mathcal{R} = \langle V, E \rangle$. The road constraints then become as described in Equation 4. Equation 5 summarizes the optimization.

$$\text{Road Constraints} \quad \forall i : K_i \in \mathcal{R} \tag{4}$$

$$\begin{aligned} &\text{Minimize} && \sum_{\forall i} (\|K_i, C_{i,i}\| + \sum_{j \in V_i} \|K_i, C_{i,j}\|), i \neq j \\ &\text{Subject to} && \forall i, v_i^j : \|C_{i,j}, K_i\| - e_i \leq 0, \\ &&& \forall i, v_i^j : k_j^x + (d_i^j * \cos \phi_i^j) = k_i^x, \\ &&& \forall i, v_i^j : k_j^y + (d_i^j * \sin \phi_i^j) = k_i^y, \\ &&& \forall i : K_i \in \mathcal{R} \\ &\text{Solving for} && \forall i : K_i \\ &\text{With parameters} && \forall i, j : C_{i,j}, \forall i : e_i, \forall i, v_i^j : \phi_i^j, \forall i, v_i^j : d_i^j, \mathcal{R} \end{aligned} \tag{5}$$

IV. Evaluation

Our evaluation focuses on CoDrive's contributions. Therefore, we omit evaluation for cloud vs edge vs cloudlet tradeoffs or computer vision related aspects. These have been already studied by several previous works. In evaluating CoDrive, we focus on the following research questions, specifically how does CoDrive perform: *(RQ1)* in preserving the shape of the traffic; *(RQ2)* as the cars' GPS accuracy deviates; *(RQ3)* as the car topology deviates; *(RQ4)* as the road segment changes; *(RQ5)* as the number of lanes deviates.

Evaluation Parameters and Plan: We derived ground truth using *Google Earth*. We used Google Earth to manually mark different topologies and acquire their GPS coordinates. We strategically position placemarks relative to recognizable landmarks in the aerial view (e.g., road signs, intersection) so that the scene could be precisely reconstructed during a real world experiment. Positioning accuracy is compared across four system regimes: GPS, Map-matching, CoDrive *w/o optimization*, and CoDrive. For the baseline, we mimic GPS errors in the most realistic way possible. In particular, we introduce GPS noise to the ground truth locations based on a Rayleigh distribution, which is often used to approximate GPS errors [28]. However, it only describes the noise magnitude. We derive erroneous locations by calculating a bearing offset using a uniform distribution between 0 and 2π. We further compare CoDrive to simple *Map-Matching*. Map-matching is a well-known technique used in all current generation navigation devices. However, most state of the art approaches require continuous GPS data of a specific vehicle's route for their inference. In constrast, CoDrive's goal is different: it utilizes snapshot data of a traffic scene among all participating vehicles encountered by a sensor-rich car, to correct their positioning. After the correction, previous dead-reckoning and map-matching approaches can be used in CoDrive, until the next correction. Thus, for a fair comparison, we use the `nearest` API of the Open-Source-Routing Machine (OSRM) [29] for the Map-Matching implementation. This reflects a reported GPS point to the nearest road segment.

Moreover we compare CoDrive with CoDrive *w/o optimization*. In the latter we *calculate* the locations of cars with respect to sensor-rich cars using the observed distance and angle information. We do this to demonstrate the effect of a sensor-rich car's positioning error. For example, if the positioning of a sensor-rich car is precise, then the calculated location should be highly accurate as well. Therefore, the optimization should not be needed. However, it is unclear how accurate a sensor-rich car can be in the real-world, at least not until they are widely adopted. In case, when multiple observations from different sensor-rich cars are received, we use their geometric median as the calculated location. Furthermore, this corresponds to the scenario, where none of the legacy cars in a scene participate. We also use CoDrive *w/o optimization* as the seed to accelerate our optimization solver.

Most of our evaluation experiments are conducted with real cars on real roads. We block different road segments within our institution for a few hours over weekends and rented cars which we positioned according to different topologies. We designate one car as the sensor-rich car with our custom kit on its roof, while the rest act as legacy cars. Distance and bearing information are real measurements captured from our sensor-rich car. Ideally, the GPS measurements for the baseline should be derived from GPS receivers placed within the vehicles. We do that for the only feasible scenario, which is within the roads of our own institution (see *REAL* in Figure 10b). However, to evaluate CoDrive in different GPS accuracy scenarios we would need to coordinate all cars to such environments. This entails various practical challenges. Therefore, we simulate GPS errors as described earlier. In all real world experiments, the distance and bearing are derived from our emulated sensor-rich car described in Section II.

For simulations, GPS ground truth of all cars is again derived from the placemarks. The cars' GPS positioning error is simulated with a Rayleigh distribution as before. The distance

and angle between a sensor-rich and legacy car are calculated using trigonometry between the simulated GPS locations of the cars. Simulation experiments are leveraged to evaluate CoDrive in scenarios with more than one sensor-rich car. Simulation experiments are indicated with a star (*).

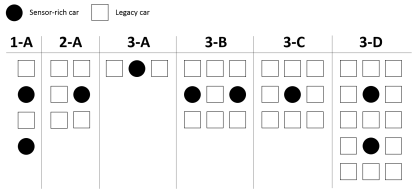


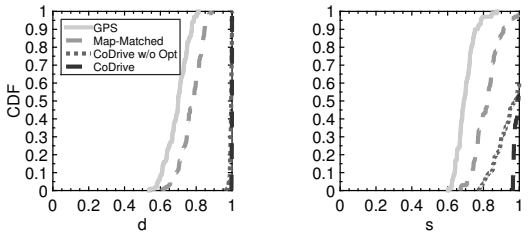
Fig. 7: Evaluation Car Topologies

We perform experiments across a varying number of **lanes** and **topologies**. The schematics of our topologies are provided in Figure 7. We further parametarize the mean and standard deviation of the introduced GPS distance error for legacy cars using μ_l, σ_l and sensor-rich cars using μ_s, σ_s.

Challenges in a Complete Real World Evaluation: Ideally, we wish to perform all experiments in the real world. Unfortunately, we face several practical hurdles. For example, to evaluate CoDrive on Topology 3-D, requires coordinating 15 cars on the road. Similarly, at least two cars need to act as sensor-rich cars and constantly moved, a time-consuming endeavor. Furthermore, repeating every experiment safely on different roads with varying number of lanes is a daunting task. Each experiment run is time-consuming, primarily because the ground truth for each car needs to be marked at the physical location and then cars need to be moved to collect the measurements. This requires blockage of the selected roads for several hours. We could not conduct experiments in an empty parking lot due to the Map-Matching baseline, which only maps a GPS location to a real-road segment (unless virtual roads are created). We also face several legal issues, including ones within the boundaries of our institution. On the other hand, relying only on the real GPS values collected when performing the experiments within our institution, would prevent us from demonstrating CoDrive's performance under different conditions. In other words, an ideal evaluation is impractical in our case, whereas a practical real-world evaluation would be incomplete. In our work we strike a balance between the two to represent different interesting scenarios and draw a more complete picture. To achieve that, we simulate GPS errors [28] to represent notable scenarios.

Evaluation Results: First, we evaluate CoDrive's accuracy in preserving the shape of traffic. This is the most important aspect of CoDrive since if it can reconstruct the traffic's geometry, it would allow for lane-level inferences. We use *procrustes shape analysis*to measure the similarity between the optimized and the original traffic view. In procrustes, a shape is represented as a graph. Procrustes finds a transformation between a source and a target shape. In our case, traffic is converted to a shape. A shape is constructed using cars as nodes and the distance/angle between them as edges. We use the ground truth car locations as the target shape. The source shape consists of the estimated locations of cars by different approaches. We only show results for topology 3-D due to its highest density of cars. The results for other topologies are indifferent, hence not shown. For these experiments we fix $\mu_l = 6m$, $\sigma_l = 2.5m$, and $\mu_s = 3m$, $\sigma_s = 1m$.

We use two standardized metrics to evaluate shape preservation: (a) a similarity measure d and, (b) a scaling measure s. d captures how similar two shapes are, which is derived from rotation/translation of a source shape to a target shape. The more similar the two shapes are, the higher the similarity. However, this metric is scale agnostic, that is, a circle with radius 1 and another with 1000 have the maximum similarity ($d = 1$). To address this, we introduce a scale measure s, which captures the scale factor after the transformations between the two shapes. Figure 8 depicts a single trial of the experiments. *GPS* looks haphazard as expected, while *Map-Matching* projects all GPS locations on a straight line (the center of the nearest road segment), retaining no lane-level information (see Figure 8c). Figure 9a and 9b quantify the same visual, across 100 trials, showing the similarity (d) and scale (s) across various approaches. The cdf captures cumulative values of d and s. Evidently, CoDrive preserves the traffic shape, making it applicable for lane-level positioning.



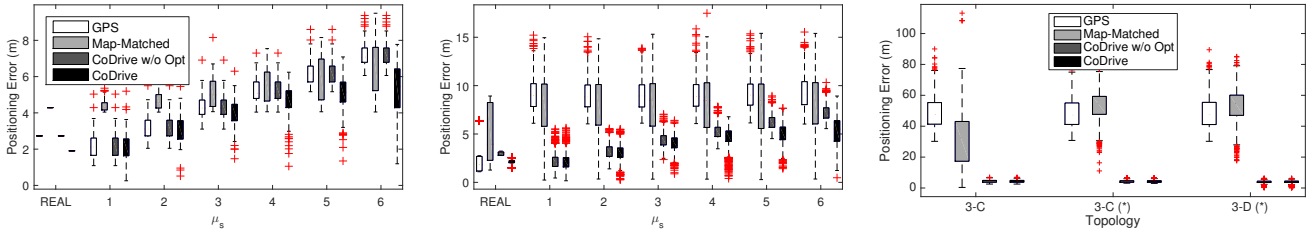
(a) Shape Similarity. (b) Scale Similarity.

Fig. 9: Accuracy in retaining the shape of the traffic.

To answer RQ2, we vary the GPS error (μ_s) for sensor-rich cars from $1-6m$, keeping $\sigma_s = 1m$; we fix the GPS error for legacy cars ($\mu_l = 6m$, $\sigma_l = 2.5m$). We also experiment with GPS measurements collected from smartphones placed in cars during the real experiment, referred to as *REAL*. Figures 10a and 10b illustrate the performance of CoDrive for sensor-rich and legacy cars respectively. CoDrive consistently outperforms both baselines *GPS* and *Map-Matched*, for both the sensor-rich and legacy cars. As expected, for smaller GPS error in sensor rich cars, CoDrive shows marginal improvement over *w/o opt* approach. However, as the error degrades, CoDrive tends to outperform the others. In particular, for sensor-rich cars, CoDrive can reduce the cumulative GPS error from 2.73m down to 1.9m in the *REAL* case, and from 7.26m to 5.20m when $\mu_s = 6m$—a 28% and 30% reduction in error respectively. For legacy cars, CoDrive brings the GPS error down from 9.3m to 5.2m when $\mu_s = 6m$ and from 9.13m down to 2.15m when $\mu_s = 1m$—a 66% and 76% reduction in error respectively. In cases, when sensor-rich car errors are very high ($\mu_s > 6m$), CoDrive continues to reduce errors but does not guarantee lane-level positioning. Note also, that

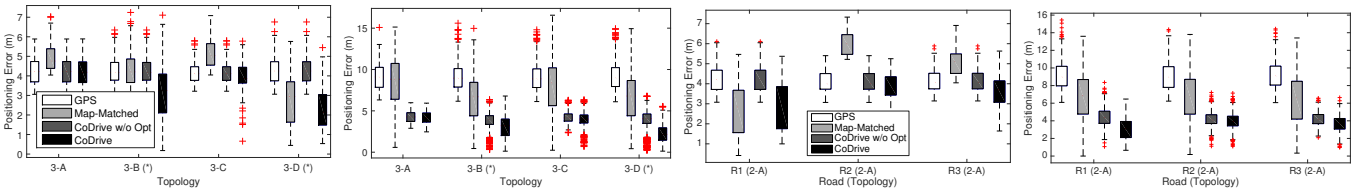
(a) Ground Truth (b) GPS (c) Map-Matched (d) CoDrive w/o opt (e) CoDrive

Fig. 8: Traffic shape preservation with different positioning approaches. CoDrive can reconstruct the aerial view of the traffic.



(a) Sensor-rich Cars ($\mu_s = 1..6$) (b) Legacy Cars ($\mu_s = 1..6$) (c) Legacy Cars ($\mu_l = 30m$, $\sigma_l = 15m$)

Fig. 10: CoDrive outperforms others as cars' accuracy degrades (10a, 10b), with key benefits in downtown-like scenarios (10c).



(a) Sensor-rich Cars (Topology) (b) Legacy Cars (Topology) (c) Sensor-rich Cars (Road) (d) Legacy Cars (Road)

Fig. 11: CoDrive outperforms others as topologies change (11a, 11b), and as road segments change (11c, 11d).

due to the clear aerial visibility and GPS line-of-sight in the roads within our institution, the *REAL* GPS positioning error—aquired by phones in vehicles—is low to begin with. This prevents us from appreciating the benefits of CoDrive in more challenging scenarios which we address by simulating GPS.

The previous experiments describe highway scenarios in a clear line-of-sight. Next, we evaluate CoDrive in a downtown scenario where GPS errors degrade between 30-50m. To approximate realistic GPS errors we fix $\mu_l = 30m$, $\sigma_l = 15m$, and $\mu_s = 3m$, $\sigma_s = 1m$. We perform the experiment on three different topologies: 3C, 3C(*), 3D(*). Figure 10c captures the accuracy of CoDrive in such extreme environments. CoDrive achieves a 91.14%, 91.19% and 92.03% reduction in error respectively which translates to a redaction in error by a factor of 11.3x, 11.3x and 12.5x respectively. Note that this reduction is for all the 8 legacy cars in 3-C and 13 in 3-D – upon a mere encounter with a sensor-rich car.

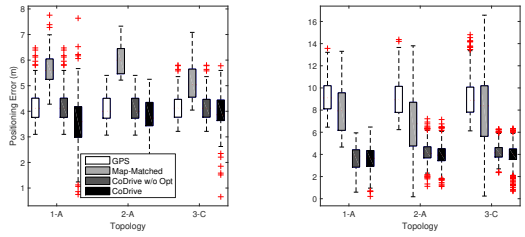
To answer RQ3, we fix GPS errors to $\mu_s = 3m, \sigma_s = 1m$ for sensor-rich cars and $\mu_l = 6m, \sigma_l = 2.5m$ for legacy cars. We use all three-lane topologies. 3-A and 3-C are real world experiments, while 3-B and 3-D are simulations. Figures 11a and 11b summarize our results. We found that CoDrive again outperforms *GPS* and *Map-Matched* in all scenarios. In addition, it remains unaffected to topology changes.

To answer RQ4, we experiment with the 2-A topology on different road segments inside our institution's campus. Ideally, the accuracy should be invariant of road conditions. However, due to different background and visual features, cloudlet computations (distance/angle) can get affected (Section II). We fix GPS error to $\mu_s = 3m, \sigma_s = 1m$ for sensor-rich cars and $\mu_l = 6m, \sigma_l = 2.5m$ for legacy cars. Figures 11c and 11d document our results. Incidentally, *Map-Matched* does well in estimating the sensor-rich car's location on road R1 (Figure 11c). This happened because the only one sensor-rich car's position and its *Map-Matching* projection coincided on the right lane. Clearly, the same is not true for the other road segments and the cars in the left lane. CoDrive remains robust in all situations.

Lastly, to answer RQ5, we design experiments for the 1-A, 2-A and 3-C topologies. We fix the GPS error to $\mu_s = 3m, \sigma_s = 1m$ for sensor-rich cars and $\mu_l = 6m, \sigma_l = 2.5m$ for legacy cars. As illustrated in Figures 12a and 12b, the number of lanes does not negatively impact CoDrive.

V. Related Work

Outdoor localization has been extensively studied. Map-matching [30], Kalman-filter and sensor fusion [31], visual odometry [32], and static landmark-based dead reckoning [33] are among the most commonly used techniques. Nevertheless, they are inadequate for next-gen applications. Differential GPS



(a) Sensor-rich Cars (b) Legacy Cars

Fig. 12: CoDrive remains robust to lane count.

[34] is a technology which addresses this issue to some extent. Unfortunately, it performs poorly in urban canyons and requires a custom receiver – hindering its widespread deployment. CoDrive, identifies a unique error correction opportunity in urban environments, where most sensor-rich cars are expected to be abundantly available in the near future.

Other works have focused on vision aided navigation. Mobileye [35] uses a forward-looking camera for advanced driver assistance systems (ADAS). SignalGuru [36] uses a phone camera and GPS to detect traffic light status and predict optimal speed for maximum fuel savings. [37] uses a phone camera and crowdsourcing to identify scenic routes. Similarly, [38] uses phones' dual cameras to detect obstacles and alert drowsy drivers on the road. [39] estimates vehicle's lane position using aerial photographs from surveillance cameras. CoDrive achieves the same without additional infrastructure or requiring a phone to be mounted on the windshield.

VI. Conclusion

This paper presents CoDrive, a system for an open-car ecosystem, where cars collaborate to improve positioning of each other. CoDrive results in precise reconstruction of a traffic scene, preserving both its shape and size. Further, CoDrive is independent of stationary landmarks and requires no additional hardware on existing cars. CoDrive utilizes coarse GPS readings of participating cars and visual, distance, and angle information of sensor-rich cars – translating them into optimization constraints. Based on these constraints, CoDrive minimizes the cumulative positioning error of all participating cars. By leveraging inter-car collaboration, CoDrive introduces a new way of achieving better positioning in dense urban environments, where most traditional approaches struggle.

References

[1] "GPS...in city environments," 2016, https://goo.gl/Pi8869.

[2] Y. Jiang, H. Qiu, M. McCartney, G. Sukhatme, M. Gruteser, F. Bai, D. Grimm, and R. Govindan, "Carloc: Precise positioning of automobiles," in *SenSys*. ACM, 2015.

[3] M. Modsching, R. Kramer, and K. ten Hagen, "Field trial on gps accuracy in a medium size city: The influence of built-up," in *WPNC*, 2006.

[4] L. Nunes and M. A. Recarte, "Cognitive demands of hands-free-phone conversation while driving," *Transportation Research Part F: Traffic Psychology and Behaviour*, 2002.

[5] S. T. Iqbal, E. Horvitz, Y.-C. Ju, and E. Mathews, "Hang on a sec!: effects of proactive mediation of phone conversations while driving," in *CHI*. ACM, 2011.

[6] J. M. Clanton, D. M. Bevly, and A. S. Hodel, "A low-cost solution for an integrated multisensor lane departure warning system," *ITS*, 2009.

[7] "The 'connected car' is creating a massive new business opportunity for auto, tech, and telecom companies," 2016, https://goo.gl/fQNvTT.

[8] "Uber Self-Driving Cars," 2016, https://goo.gl/xPGsCg.

[9] "The AI Car Computer for Self-Driving Vehicles," 2017, https://goo.gl/EcdKd4.

[10] "Telsa Auto Pilot," 2015, https://goo.gl/ZHr86d.

[11] "Ford targets fully autonomous vehicle..." 2016, https://goo.gl/n5PXyd.

[12] D. Chen, K.-T. Cho, S. Han, Z. Jin, and K. G. Shin, "Invisible sensing of vehicle steering with smartphones," in *MobiSys*. ACM, 2015.

[13] J. C. McCall and M. M. Trivedi, "An integrated, robust approach to lane marking detection and lane tracking," in *IV*. IEEE, 2004.

[14] S. Maldonado-Bascon, S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno, and F. López-Ferreras, "Road-sign detection and recognition based on support vector machines," *ITS*, 2007.

[15] "Average age of light vehicles in the U.S. rises to 11.5 years," 2015, http://goo.gl/vhBuVW.

[16] "How Google's self-driving cars detect and avoid obstacles," 2016, https://goo.gl/bUA781.

[17] D. Li, T. Bansal, Z. Lu, and P. Sinha, "Marvel: multiple antenna based relative vehicle localizer," in *MobiCom*. ACM, 2012.

[18] M. Obst, N. Mattern, R. Schubert, and G. Wanielik, "Car-to-car communication for accurate vehicle localization – the covel approach," in *SSD*. IEEE, 2012.

[19] "Cloud vs edge in an iot world," 2016, https://iotworldnews.com/2016/04/cloud-vs-edge-in-an-iot-world/.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[22] G. Nebehay and R. Pflugfelder, "Clustering of Static-Adaptive correspondences for deformable object tracking," in *CVPR*. IEEE, 2015.

[23] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *ICCV Workshops*, 2013.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *IJRR*, 2013.

[25] S. T. Birchfield and S. Rangarajan, "Spatiograms versus histograms for region-based tracking," in *CVPR*. IEEE, 2005.

[26] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of rgb camera with velodyne lidar."

[27] "Machine learning in navigation devices: detect maneuvers using accelerometer and gyroscope," 2015, https://goo.gl/gqkkSR.

[28] "GPS horizontal positioning accuracy," 2016, https://goo.gl/vdrNr6.

[29] D. Luxen and C. Vetter, "Real-time routing with openstreetmap data," in *SIGSPATIAL*. ACM, 2011.

[30] J. S. Greenfeld, "Matching gps observations to locations on a digital map," in *Transportation Research Board 81st Annual Meeting*, 2002.

[31] M. S. Grewal, *Kalman filtering*. Springer, 2011.

[32] M. Agrawal and K. Konolige, "Real-time localization in outdoor environments using stereo vision and inexpensive gps," in *ICPR*. IEEE, 2006.

[33] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low-cost outdoor localization for very small devices," *IEEE Pers. Commun.*, 2000.

[34] B. W. Parkinson and P. K. Enge, "Differential gps," *Global Positioning System: Theory and applications.*, 1996.

[35] "Mobileye: machine vision technology for mono-cameras," 2017, http://www.mobileye.com/.

[36] E. Koukoumidis, L.-S. Peh, and M. R. Martonosi, "Signalguru: leveraging mobile phones for collaborative traffic signal schedule advisory," in *MobiSys*. ACM, 2011.

[37] S. Morishita, S. Maenaka, D. Nagata, M. Tamai, K. Yasumoto, T. Fukukura, and K. Sato, "Sakurasensor: quasi-realtime cherry-lined roads detection through participatory video sensing by cars," in *UbiComp*. ACM, 2015.

[38] C.-W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, M. Montes-de Oca, Y. Cheng, M. Lin, L. Torresani *et al.*, "Carsafe app: alerting drowsy and distracted drivers using dual cameras on smartphones," in *MobiSys*. ACM, 2013.

[39] B. Dorj and D. J. Lee, "A precise lane detection algorithm based on top view image transformation and least-square approaches," *Journal of Sensors*, 2015.