Pervasive and Mobile Computing 26 (2016) 35–56

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Robust and ubiquitous smartphone-based lane detection☆

Heba Aly[a,*], Anas Basalamah[b], Moustafa Youssef[c,1]

[a] Department of Computer Science, University of Maryland, USA
[b] Comp. Eng. Department & KACST GIS Tech. Innov. Ctr., Umm Al-Qura University, Saudi Arabia
[c] Wireless Research Center, E-JUST, Egypt

ARTICLE INFO

Article history:
Available online 6 November 2015

Keywords:
Lane detection
Smartphone-based localization
Crowd-sensing
Smartphone-based sensing
Energy-efficient systems

ABSTRACT

Lane-level positioning is required for several location-based services such as advanced driver assistance systems, driverless cars, predicting driver's intent, among many other emerging applications. Yet, current outdoor localization techniques fail to provide the required accuracy for estimating the car's lane.

In this paper, we present *LaneQuest*: an accurate and energy-efficient smartphone-based lane detection system. *LaneQuest* leverages hints from the ubiquitous and low-power inertial sensors available in commodity off-the-shelf smartphones about the car's motion and its surrounding environment to provide an accurate estimate of the car's current lane position. For example, a car making a u-turn, most probably, will be in the left-most lane; a car passing by a pothole will be in the pothole's lane; and the car angular velocity when driving through a curve reflects its lane. Our investigation shows that there are amble opportunities in the environment, i.e. lane "anchors", that provide cues about the car lane. To handle the ambiguous location, sensors noise, and fuzzy lane anchors; *LaneQuest* employs a novel probabilistic lane estimation algorithm. Furthermore, it uses an unsupervised crowd-sourcing approach to learn the position and lane span distribution of the different lane-level anchors.

Our evaluation results from implementation on different Android devices and driving traces in different cities covering 260 km shows that *LaneQuest* can detect the different lane-level landmarks with an average precision and recall of more than 91%. This leads to an accurate detection of the exact car lane position 84% of the time, increasing to 92% of the time to within one lane. This comes with a low-energy footprint, allowing *LaneQuest* to be implemented on the energy-constrained mobile devices.

1. Introduction

Recently, there has been a growing number of location-based services (LBS) that require knowledge of the car's lane position including advanced driver assistance systems (ADASs) [1], autonomous cars (e.g. the Google driverless car [2]), lane-based traffic estimation, electronic toll fee collection [3], predicting driver's intent [4,5], among others. However, current state-of-the-art outdoor vehicular navigation systems can only provide an accuracy of about 10 m in urban environments [6]. Hence, they fail to provide an estimate of the vehicle's exact lane (Fig. 1).

☆ An early version of this paper appeared in the proceedings of the IEEE PerCom'15 (Aly et al., 2015).

* Corresponding author.
E-mail addresses: heba@cs.umd.edu (H. Aly), ambasalamah@uqu.edu.sa (A. Basalamah), moustafa.youssef@ejust.edu.eg (M. Youssef).

1 Moustafa Youssef is currently on sabbatical from Alexandria University, Egypt.

http://dx.doi.org/10.1016/j.pmcj.2015.10.019
1574-1192/

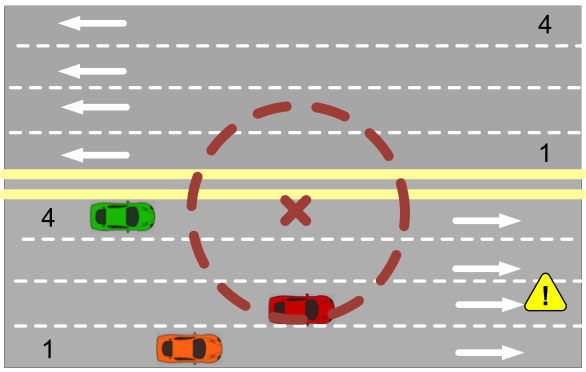


Fig. 1. Current outdoor localization technologies fail to provide enough accuracy to estimate the car lane position. The '×' mark denotes the GPS position and the circle the associated error. While the red car is moving in the 2nd lane, an error around 3 m moves its estimate to the 4th lane. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

A number of systems were proposed to provide lane-level localization accuracy [7–12]. However, these systems require special sensors to be installed on all vehicles (e.g. the RF sensors in [11]) and/or an expensive calibration phase (e.g. [7–10]), limiting their ubiquitous deployment. Computer vision based techniques, e.g. [12], use a camera to detect the lane markings. However, using an image processing solution raises accuracy challenges when road markings are unclear, line-of-sight is obstructed, and/or in bad weather conditions (e.g. raining). It also requires extensive energy and processing power from commodity smartphones.

In this paper, we present *LaneQuest*; a system that leverages the ubiquitous sensors available in commodity smartphones to provide an accurate and energy-efficient estimate of the car current lane. Starting from an ambiguous coarse location estimate, e.g. reported by the GPS, *LaneQuest* leverages driving events detected by the phone sensors to reduce this ambiguity. Specifically, *LaneQuest* uses the low-energy inertial sensors measurements to recognize unique motion events while driving such as changing the lane, turning right, or passing over a pothole. These events or "lane anchors" provide hints about the car current lane. For example, a car making a left turn most probably will be in the left-most lane; Similarly, potholes typically span only one lane, allowing detecting the lane of cars that pass by them. *LaneQuest* uses a crowd-sensing approach to detect a large class of lane anchors as well as their positions through the road network and the lanes they span, exploiting them as opportunities for reducing the ambiguity in lane estimation.

To address the sensors' noise, location ambiguity, and error in anchors location estimation; *LaneQuest* models the lane detection problem as a Markov-localization problem that combines the vehicle's motion events (such as changing lanes) with lane anchor detection in a unified probabilistic framework. We have implemented *LaneQuest* on different Android devices and evaluated it using driving experiments at different cities covering more than 260 km. Our results show that *LaneQuest* can detect the different lane anchors with an average precision and recall of 93% and 91% respectively. This leads to accurately detecting the car lane more than 84% of the time, increasing to 92% to within one lane error. Moreover, *LaneQuest* has a low-energy profile when implemented on top of different localization techniques.

In summary, our main contributions are summarized as follows:

- We present the *LaneQuest* architecture: an energy-efficient crowd-sensing system that leverages the sensed lane-anchors along with the vehicle's dynamics to provide an accurate robust estimate of the car's current lane position without any prior assumption on her starting lane position.
- We provide the details of a unified probabilistic framework for robust detection of the vehicle's driving lane position.
- We propose a crowd-sensing approach for detecting the road and lane position for different types of lane anchors. The proposed technique captures the inherent ambiguity in the crowd-sensing process.
- We implement *LaneQuest* on Android phones and evaluate its performance and energy-efficiency in different cities using different smartphone devices.

In an earlier work [13], we proposed a preliminary version of the *LaneQuest* system presented in this paper. However, in this paper we have extended the work in [13] significantly. Specifically, we propose a new lane estimation algorithm (*minErr*) to provide more robust and accurate estimates. The proposed method enhances the lane estimation accuracy by more than 20% compared to the method in [13]. In addition, it reduces the errors incurred during the transient period significantly. We also add a new curvature estimation method. The new method works with phones that do not support the gyroscope sensor and enables our system to cope with different heterogeneous devices. Moreover, we propose a new lane-change algorithm detection. The proposed algorithm improves the accuracy of detecting both left and right lane changes by 26.8% and 7.3% respectively as compared to the conference version.

The rest of the paper is organized as follows: we discuss the related work in Section 2. Section 3 presents an overview of the system architecture. Sections 4 and 5 give the details of the *LaneQuest* system and event detection framework. We discuss the different aspects of the system in Section 6. Section 7 provides our evaluation of *LaneQuest*. Finally, we conclude the paper and give directions for future work in Section 8.

Table 1
Comparison between *LaneQuest* and other smartphone-based lane estimation techniques.

Name	Sensor(s)	Power usage	Run time	Pitfall(s)	Constraint(s)	Lane detection accuracy
LaneQuest	Inertial sensors	Low	Fast (O(ms))	No error resetting if the user is driving straight on a straight road with no anchors.	Coarse-grained location available (e.g. GPS or Network-based).	High (∼84% exact lane and ∼92% within one lane error)
Ren et al. [12]	Camera	High	Medium (O(s))	Bad weather conditions (e.g. snow, rain) lighting condition (e.g. sun glare, headlight glare, shadows from nearby buildings, etc.) environmental noise (e.g., faded lane marks, surrounding objects like buildings, parked cars, etc.)	Good lighting conditions – clear visible straight lane marking – high quality camera.	High (90% with *visible clear lane marking* – fail otherwise)
Chanawangsa et al. [16]	Camera	High	Medium (∼100 ms)	Bad weather conditions (e.g. snow, rain) lighting condition (e.g. sun glare, headlight glare, shadows from nearby buildings, etc.) environmental noise (e.g., faded lane marks, surrounding objects like buildings, parked cars, etc.)	Good lighting conditions – multi-core phone – high quality camera.	High (90% with *visible clear lane marking* – fail otherwise)

2. Related work

Through this section, we discuss the different lane-level localization techniques and previous techniques that use inertial sensors for sensing different road parts or driving behavior.

2.1. Lane determination

Current state-of-the-art localization techniques can only provide location estimates with an average accuracy around 10 m [6], which is not suitable for lane-level localization. To overcome this, researchers proposed different techniques that are based on using special sensors and/or smartphone sensors [8,9,14,15,12,16,17].

2.1.1. Techniques that use special sensors

To overcome the Global Navigation Satellite Systems (GNSS) inaccuracies, researchers proposed to fuse it along with other sensors to provide more accurate lane-level location accuracy [8,9,14,15]. For example, in [8,9] authors fuse a high-accuracy GNSS receiver, an odometer, and a gyroscope, along with an enhanced digital map that describes the road geometry using a particle filter-based algorithm to position the user based on a combined GNSS-dead-reckoning approach and map matches her to her lane. Similarly, in [10] authors fused an L1-GPS device, camera and an enhanced digital map that stores lane markings information using a dynamical Kalman filter with map matching to get an accurate user position. Also, in [17], authors fused the LIght Detection And Ranging (LIDAR) signal with the vision data to detect the driving lane for autonomous vehicle navigation. MARVEL [11] provides relative lane localization using an antenna diversity-based solution by installing four special RF transceivers on each car in addition to using the GPS and inertial sensors. It enables a vehicle to determine the other vehicles' relative position (in the same, right or left lane), and which vehicle is ahead of the other. All these techniques require special hardware, sometimes with ubiquitous deployment, in order to function. This limits their applicability on a large scale. *LaneQuest*, on the other hand, provides an accurate lane estimate using only energy-efficient inertial sensors widely available on off-the-shelf smartphones, eliminating the need for any special devices to be installed on the vehicle or an expensive pre-calibrated enhanced digital map.

2.1.2. Techniques that use smartphone sensors

Table 1 shows a comparison of *LaneQuest* with the other smartphone-based lane estimation techniques. In [12], authors proposed to use the iPhone camera to detect the lane markings. Chanawangsa et al. [16], proposed to utilize the multi-core capabilities in new smartphones to improve the lane-detection accuracy and latency. However, using cameras for lane detection is highly susceptible to errors due to various factors such as lighting condition (e.g. night time, sun glare, headlight glare, shadows from nearby buildings, etc.), bad weather conditions (e.g. snow, rain), and other environmental noise (e.g., faded lane marks, surrounding objects like buildings, parked cars, etc.). Moreover, cameras have high energy requirements for the limited phone battery (as we quantify in Section 7).

LaneQuest, on the contrary, provides an accurate lane estimate using only energy-efficient inertial sensors widely available on off-the-shelf smartphones. Also, it employs a probabilistic framework that takes into account possible uncertainties due to sensor noise and crowd-sourcing, providing a robust lane estimate.

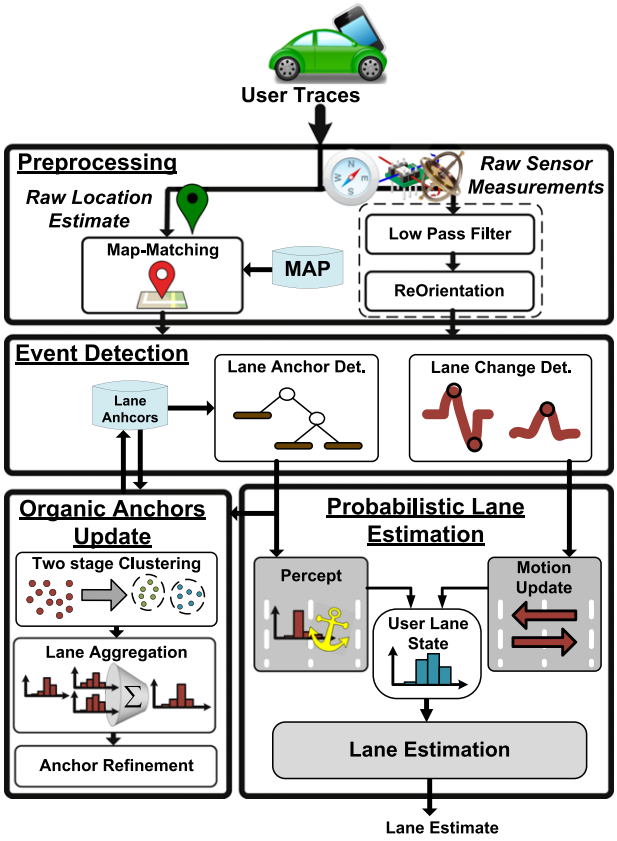


Fig. 2. *LaneQuest* system architecture. *LaneQuest* predicts the car current lane using a probabilistic approach by fusing knowledge of the car lane changes and a repository of lane-level anchors. Crowd-sourced traces are also used to detect new anchors and identify their lane position in an organic way.

2.2. Road sensing and driving behavior detection

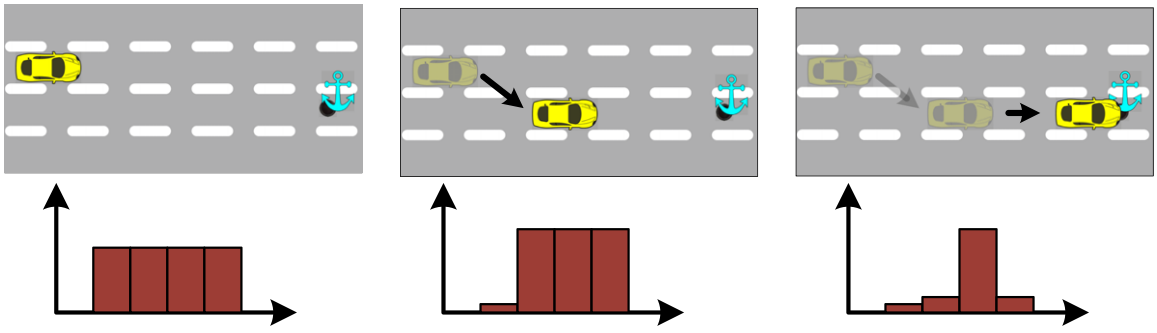
Inertial sensors have been used in literature for detecting driving behavior [18–20], identifying map semantics [21], and monitoring road problems [22,23]. For example, in [18,20] authors used inertial sensors to detect the driving quality of the driver. They identified driving patterns events like lane-changing and acceleration/deceleration and rated the driver according to the frequency and suddenness of these events. In [19], authors used inertial sensors and an external accelerometer installed on the car to sense the vehicle dynamics when moving over turns to detect the driver's phone usage (whether the phone is on the left or right side of the vehicle). The goal was to disallow using the phone while driving. *LaneQuest* leverages similar events for determining the current lane with extensions to separate close events, such as making a turn or moving on a curve, for more robust and accurate lane localization.

In [21], authors inferred various map-semantics to enrich digital maps such as tunnels, roundabouts, and bridges among others using smartphone's inertial sensors and cellular-information. In [24], authors used the cellphone accelerometer to detect the potholes without separating them from normal traffic calming devices, e.g. bumps. The Pothole Patrol [22] and the Nericell [23] systems, on the other hand, use a 3-axis accelerometer and GPS to detect potholes along the road and apply a series of filters on the acceleration to separate between potholes and others like bumps and expansion joints. They assume that the locations of intended speed bumps are *known* to separate between them and potholes. Both Nericell and Pothole Patrol use *external* sensor chips which have higher sampling rates and lower noise compared to chips available on typical smartphones in the market.

Compared to these systems, *LaneQuest* uses the *cheap noisy inertial sensors* in standard cell phones to detect driving patterns like changing lanes or road semantics like tunnels. More importantly, it identifies more fine-grained anchors ***at the lane-level***, e.g. instead of just detecting one anchor for the tunnel, it detects different anchors for the lanes inside the tunnel. In addition, it uses an **unsupervised** crowd-sourcing approach to learn the signatures of these lane-level anchors.

3. System overview

Fig. 2 shows an overview of the *LaneQuest* system architecture. *LaneQuest* estimates the car's lane position using the inertial sensors available on a cell-phone attached to the vehicle's windshield or a dashboard-mount. It leverages the



(a) Unknown lane position. (b) Most probably not at the fourth lane. (c) Most probably at the second lane.

Fig. 3. The probabilistic lane estimation basic idea: At the beginning, the car's lane is unknown. Then, as the car moves to the adjacent right lane the distribution moves to the right since, most probably, it is not at the left-most lane anymore. Finally, as the car encounters a landmark, its lane is mostly known as the landmark's lane.

vehicle dynamics (e.g. changing lanes) and detected lane anchors in a probabilistic Markov framework to estimate the vehicle's current lane. The system has four main components: the Preprocessing module, the Event Detection module, the Probabilistic Lane Estimation module, and the Organic Anchors Update module. In this section, we give an overview of each of these modules.

3.1. Preprocessing module

This module is responsible for preprocessing the raw input sensors and location data to reduce the noise effect. *LaneQuest* collects time- and location- stamped measurements from the energy-efficient inertial sensors in the cell-phone. These include the accelerometer, gyroscope and magnetometer. To handle the noise in the sensors readings, we apply a local weighted low-pass regression filter [25]. In addition, we also transform the sensors readings from the mobile coordinate system to the car coordinate system leveraging the inertial sensors [26]. After this transformation, the sensors y-axis points to the car direction of motion, x-axis to the left side of the car, and z-axis is perpendicular to Earth (pointing to the car ceiling). We refer the reader to [26] for more information about sensor-measurements re-orientation.

For location information, *LaneQuest* does not require a specific localization technique; it can leverage GPS, network-based localization techniques [27–29] or other more accurate and energy-efficient GPS-replacement techniques, e.g. [6]. To further enhance the input location accuracy, we apply map matching [30] to align the car's location estimates to the road network.

3.2. Event detection module

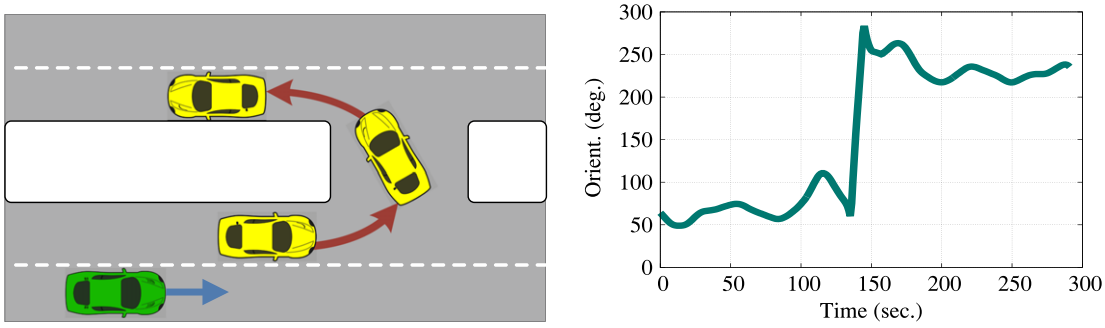
There are many driving patterns that can give cues for the vehicle's current lane based on their unique signature on the different phone senors. For example, when a car moves to the adjacent lane to the right, the car is with high probability not in the left-most lane (Fig. 3). This "lane change event" can be detected by the phone inertial sensors (using the *Lane Change Detection* sub-module) and used to reduce the ambiguity in the car's current lane.

Similarly, when making a u-turn, the car is most probably at the left-most lane before and after the u-turn. Therefore, noting that the car's direction changes by around $\pm 180^\circ$ when making a u-turn, which can be captured using the cellphone's orientation sensor (Fig. 4) using the *Lane Anchor Detection* module, this "u-turn anchor" hint is used by *LaneQuest* to reduce the ambiguity of the vehicle's current lane.

LaneQuest differentiates between two types of lane anchors: **Bootstrap anchors** and **Organic anchors**. *Bootstrap anchors* have a clear *pre-known* lane distribution across the road. For example, stopping a car occurs in the right-most lane; a u-turn is initiated in the left-most lane, and a right-turn happens with high probability in the right-most lane. On the other hand, *organic anchors* have unique signatures across the different lanes but their lane distribution *cannot be pre-known* and need to be learned. For example, a pothole can be detected by the phone sensors as we show in Section 4. However, we do not know a priori in which lane this pothole is located. *LaneQuest* uses an unsupervised crowd-sourced approach to capture these anchors and identify their lanes distribution. The details of operation of this module are discussed in Section 5.

3.3. Probabilistic lane estimation module

To achieve robust and accurate lane estimates based on the noisy inertial sensors measurements, the ambiguous car locations, human driving anomalies, and fuzzy lane anchor locations; *LaneQuest* uses a probabilistic estimation technique. Specifically, our lane estimation technique is based on Markov Localization [31,32], which is known in the robotics domain for addressing the problem of state estimation from noisy sensors data. Instead of maintaining a single hypothesis about the robot location, Markov localization uses a probabilistic framework to maintain a probability density over the set of



(a) A car doing a u-turn will be at the left-most lane with high probability.

(b) U-turns cause a change around 180° in the orientation sensor.

Fig. 4. The phone orientation sensors can detect a u-turn, which in turn gives a better idea about the car current lane.

Table 2
Notations used in the paper.

Notation	Definition
n_t	Number of road's lanes at time t.
ℓ_t	Car's actual lane position at time t ($\ell_t \in \{1, \ldots, n\}$).
L_t	A discrete random variable that expresses the car's lane at time t.
$Bel(L_t)$	Car's lane belief at time t (probability mass function) over all the lanes.
e_t	The event detected by *LaneQuest* at time t (lane change or passing by a lane-anchor)
m_t	The detected lane change event (i.e. lane changes to the right or left)
m^l	Left lane change event
m^r	Right lane change event
m^0	No lane change event
a_t	The detected lane anchor event at time t (e.g. pothole or a u-turn).
s_t	The estimated lane position at time t using our probabilistic lane estimation algorithm.
α_T	Normalization factor for the perception update at time t.
$lErr(L_t = \ell)$	The error score for lane ℓ at time t where $\ell \in \{1, \ldots, n\}$.

possible locations. Such a density can have an arbitrary form representing various position beliefs, including multi-modal distributions. Markov localization can deal with ambiguous situations and it can re-localize the robot position in the case of localization failures. The basic assumption in Markov localization is that the current state, i.e. the current robot location, captures the entire movement history (Markov assumption). That is, the current position is the only state in the environment which systematically affects the sensors readings.

Accordingly, *LaneQuest* uses Markov localization to maintain a probability distribution over all possible lanes. This probabilistic representation allows it to weigh the different hypotheses and reach a more accurate lane estimate in a mathematically principled way. *LaneQuest* does not make any assumption on the starting lane position of the car. This is modeled as a uniform distribution across all lanes. Then, as the car moves on the road, any cues for the car motion (i.e. lane changes) or detected lane anchors (e.g. a pothole) are used to update this lane belief distribution (Fig. 3). For example, assuming a car is moving on a four-lane road and it made three right lane changes, each time a lane change is detected the car's lane position distribution is updated. After the third lane change, the car is at the right-most lane with high probability. Similarly, if we know that the road has a pothole at the second lane around the current car location and the car encounters it, then most probably it is at the second lane.

The details of operation of this module are discussed in Section 4.2.

3.4. Organic lane anchors updates module

This module is responsible for estimating the road location and lane distribution of organic anchors such as curves and potholes. It uses a crowd-sensing approach, where the information about the detected lane anchors by different system users is collected and processed to estimate the anchor location and lane distribution based on the reporting cars' lane distributions. The details of operation of this module are discussed in Section 5.3.

4. Probabilistic lane estimation

In this section, we provide the details of the *LaneQuest* novel probabilistic lane estimation approach. Table 2 summarizes the notations used in this section.

4.1. Model overview

- Let ℓ_t denote the actual car's lane position at time t and L_t denote the corresponding discrete random variable. ℓ_t can take values from 1 to n; where n is the number of road lanes.
- The belief about the car lane position at time t is $Bel(L_t)$. $Bel(L_t)$ is the probability mass function representing the probability distribution over the road's lanes.
- Let e_t denote the event detected at time t. The system can detect two types of events: motion events m_t (i.e. lane changes to the right or left) and lane anchor detection event a_t (e.g. pothole or a u-turn).
- Let s_t denote the estimated lane position at time t using our probabilistic lane estimation algorithm.

4.2. Probabilistic lane estimation

Our lane estimation module aims to estimate the car's lane using the detected events (motion events and anchors detection events). Since the lane belief ($Bel(L_t)$) changes only when the phone sensors detect an event, at time T, the car's lane belief will be based on all detected events till that time ($e = e_0, e_1, \ldots, e_T$). This corresponds to the posterior distribution over the road's lane conditioned on all the detected events, that is

$$Bel(L_t) = P(L_t = \ell|e) = P(L_T = \ell|e_0, \ldots, e_T). \tag{1}$$

When computing $P(L_t = \ell|e)$, we have two cases based on the two event types (motion event and anchor detection event).

4.3. Case 1: motion update

LaneQuest performs a motion update when the detected event is a motion event, i.e. lane change ($e_T = m_T$).
A car moving over the road will use the same lane till it makes a right (m^r) or a left (m^l) lane change.
Therefore, when the phone sensors detect a lane change (as in Section 5.1), the lane belief in Eq. (1) can be factorized to:

$$Bel(L_T) = P(L_T = \ell|e) = \sum_{i=1}^{n} P(L_T = \ell|e, L_{T-1} = \ell_i) P(L_{T-1} = \ell_i|e). \tag{2}$$

This is based on the theorem of Total Probability [33], where the probability that the car is at a certain lane ℓ as a result of a lane change event is mapped to the summation of the possibility of being at any previous lane position multiplied by the transition probability of moving to ℓ from this previous lane.
From the Markovian assumption, we can further simplify the term $P(L_T = \ell|e, L_{T-1} = \ell_i)$ to:

$$\begin{aligned} P(L_t = \ell|e, L_{T-1} = \ell_i) &= P(L_T = \ell|e_0, \ldots, e_{T-1}, m_T, L_{T-1} = \ell_i) \\ &= P(L_T = \ell|m_T, L_{T-1} = \ell_i). \end{aligned} \tag{3}$$

Similarly, m_T should not affect the lane position at $T-1$ in the term $P(L_{T-1} = \ell_i|e)$. Hence Eq. (2) can be written as:

$$Bel(L_T = \ell) = P(L_t = \ell|e) = \sum_{i=1}^{n} P(L_T = \ell|m_T, L_{T-1} = \ell_i) P(L_{T-1} = \ell_i|e_0, \ldots, e_{T-1}), \tag{4}$$

which can be written in a recursive form as:

$$Bel(L_T = \ell) = \sum_{i=1}^{n} P(L_T = \ell|m_T, L_{T-1} = \ell_i) Bel(L_{T-1} = \ell_i). \tag{5}$$

The probability $P(L_T = \ell|m_T, L_{T-1} = \ell_i)$ represents the motion model and it should capture the uncertainty in our sensors measurements. We model this uncertainty using the lane change event confusion matrix (Section 5.1.1). For example, if the current detected motion event is a left lane change, then $P(L_T = \ell|m_T = m^l, L_{T-1} = \ell_i)$ is calculated as:

$$P(L_T = \ell|m_T = m^l, L_{T-1} = \ell_i) = \begin{cases} P(m^l|m^l) & \ell = \ell_i + 1 \\ P(m^l|m^r) & \ell = \ell_i - 1 \\ P(m^l|m^0) & \ell = \ell_i \\ 0 & o.w. \end{cases} \tag{6}$$

where m^0 represents the no lane change event.

4.4. Case 2: perception update

The second type of events we have in our system is passing by one of the lane-anchors, i.e. $e_T = a_T$. *LaneQuest* performs a perception update when the detected event is observing an anchor ($e_T = a_T$).

In this case, Eq. (1) can be factorized using Bayes' rule to:

$$P(L_t = \ell|e) = \frac{P(a_T|e_0, \ldots, e_{T-1}, L_T = \ell)P(L_T = \ell|e_0, \ldots, e_{T-1})}{P(a_T|e_0, \ldots, e_{T-1})}. \tag{7}$$

That can be simplified based on our Markov assumption to:

$$P(L_t = \ell|e) = \frac{P(a_T|L_T = \ell)P(L_T = \ell|e_0, \ldots, e_{T-1})}{P(a_T|e_0, \ldots, e_{T-1})}. \tag{8}$$

Noting that the denominator of the last equation does not depend on L_T, we can replace it by a constant (α_T) (i.e. a normalizing factor). Therefore, Eq. (8) becomes:

$$P(L_t = \ell|e) = \alpha_T P(a_T|L_T = \ell)P(L_T = \ell|e_0, \ldots, e_{T-1}). \tag{9}$$

Again, this can be put in a recursive form as:

$$Bel(L_t = \ell) = (\alpha_T)P(a_T|L_T = \ell)Bel(L_{T-1} = \ell). \tag{10}$$

The term $P(a_T|L_T = \ell)$ represents the perception model, which is the likelihood that the lane-anchor's signature a_T would be observed if the user was actually in lane (ℓ). Two factors affect this model: whether there is actually an anchor of the detected type near the car current location and the anchor lane distribution. Therefore, we model this probability as a weighted Gaussian distribution as:

$$P(a_T|L_T = \ell) = P(\ell|a_T)\frac{1}{\sqrt{2\pi}\sigma}e^{-0.5\left(\frac{|\ell - \ell_{a_T}|}{\sigma}\right)^2} \tag{11}$$

where $|\ell - \ell_{a_T}|$ is the distance between the a_T anchor's lane position ℓ_{a_T} and the car current lane position ℓ and σ is the uncertainty in the sensors measurements and the anchor's lane position. We estimate σ as the median absolute deviation (MAD) which is a robust estimator of σ [34].

$$\sigma = 1.4826 \times \text{median}_T(|\ell - \ell_a|). \tag{12}$$

4.5. Lane estimation

LaneQuest uses the estimated lane belief $Bel(L_T = \ell)$ at time T to predict the user's lane s_T. We propose two lane-estimation methods: the *maxBel* and *minErr* methods.

Method 1: *maxBel*

The simplest method to estimate the car lane given the lane belief distribution is to select the lane with the maximum probability. More formally, the *maxBel* method estimates the car current lane (s_T) at time T as:

$$s_T \leftarrow \arg\max_{\ell} Bel(L_T = \ell). \tag{13}$$

Method 2: *minErr*

In this method, we predict the user's lane as the one that minimizes the expected lane error $lErr(L_T = \ell)$ at time T; which we estimate as follows:

$$lErr(L_T = \ell) = \sum_{j=1}^{n} |\ell - j| \times Bel(L_T = j). \tag{14}$$

Accordingly, the *minErr* method estimates the car current lane (s_T) at time T as:

$$s_T \leftarrow \arg\min_{\ell} lErr(L_T = \ell). \tag{15}$$

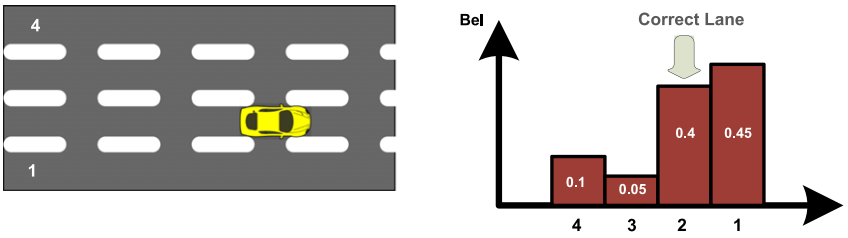
For lanes with the same expected error ($min(lErr)$), we choose the most probable lane of them.

This method improves the system's overall accuracy especially when the lane belief distribution does not have a unique high peak (Fig. 5). The intuition behind this method is that instead of selecting the lane with the highest probability, we select the lane that will lead to the lowest expected error. Since this method takes into account the probability from all the lanes believes, it leads to better performance as we quantify in Section 7.

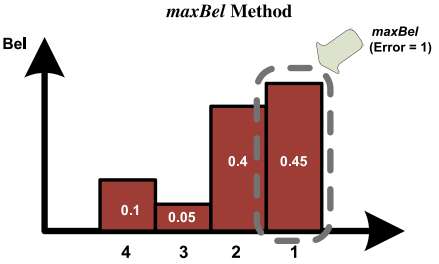
Our lane estimation algorithm is summarized in Algorithm 1 using the *minErr* method.

5. Events detection

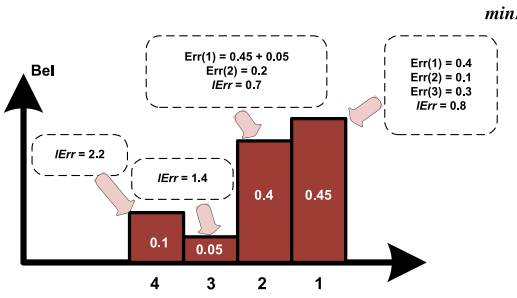
In this section, we describe how *LaneQuest* detects the motion events (i.e. lane change) and the anchor detection events.



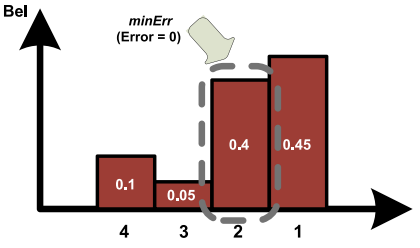
(a) A car moving on the second lane with its lane belief (*Bel*).



(b) Using *maxBel* method: We choose the lane with the maximum belief regardless from the fact that the second lane has high probability too (0.05 difference); leading to an error of one.



(c) In *minErr* method, we estimate the probable error for every lane using (Equation 14). *Err(l)* represents the probability of having an error *l* weighted by *l*.



(d) Using *minErr* method: We choose the lane with the minimum probable error; leading to zero error.

Fig. 5. Example usage of the proposed lane estimation methods: *maxBel* and *minErr*.

5.1. Lane change detection

Drivers typically change lanes for several reasons including: (a) the current lane is ending/merging, (b) the driver plans to make a turn at an upcoming intersection, or (c) the driver wants to move to a faster/slower moving lane. A number of techniques in literature proposed using the phone inertial sensors to detect the car lane change event [18,11]. The idea is that for a car to change its lane, it experiences a change in its direction (Fig. 7), which causes a rotation around the *z*-axis of the accelerometer (for the oriented phone) and affects mainly the *x*-acceleration [35]. Assuming that the vehicle is making a left-lane change (Fig. 6), then the *x*-acceleration reading first decreases to a low value and then increases back to a higher value. It also minimally affects the phone's orientation.

However, our experiments show that this *x*-acceleration pattern is not unique to a lane-change event. It can also happen in other cases when the car changes its direction, e.g. due to taking a turn or moving over a curve. This makes it harder to separate the lane-change events. To make our lane change detection more robust, we *extend previous work* to separate between lane changes and other cases using the orientation variance. The idea is that, typically, curves and turns will cause the car to have much higher variations as compared to lane-changes.

We employ a simple threshold-based method using both *x*-acceleration and orientation measurements. We identify the maximum and minimum peaks in the *x*-acceleration within a window and detect a lane change event *only if* the difference between them is high while the variance in orientation is low. The direction of the lane change is then detected based on

Algorithm 1 Lane Estimation Algorithm using minErr Method

```
for each ℓ do
    Bel(L_0 = ℓ) ← 1/n                                   ▷ Initializes the lane belief
end for
while true do
    if an anchor is detected then
        α_T ← 0
        for each ℓ do                                    ▷ Perception model
            B̂el(L_T = ℓ) ← P(a_T|ℓ)Bel(L_{T−1} = ℓ)
            α_T ← α_T + B̂el(L_T = ℓ)
        end for
        for each ℓ do                                    ▷ Normalize the lane belief
            Bel(L_T = ℓ) ← α_T^{−1} B̂el(L_{T−1} = ℓ)
        end for
    end if
    if lane change detected then
        for each ℓ do                                    ▷ Motion model
            Bel(L_T = ℓ) ← Σ_{i=1}^{n} P(ℓ|ℓ_i, m_T) B̂el(L_{T−1} = ℓ_i)
        end for
    end if
    s_T ← LaneEstimate(Bel)
end while
procedure LaneEstimate(Bel)
    for each ℓ do
        lErr(L_T = ℓ) ← 0                                ▷ Initializes the score function
    end for
    for i = 1 to ℓ do
        for j = 1 to ℓ do                                ▷ Compute the score
            lErr(L_T = i) = lErr(L_T = i) + |i − j| × Bel(L_T = j)
        end for
    end for
    S ← arg min_ℓ lErr(L_T = ℓ)
    if size(S) = 1 then                                  ▷ One lane has min. error
        return S
    else
        return arg max_ℓ S(ℓ)                            ▷ Return most-probable lane
    end if
end procedure
```

Table 3
Lane change event confusion matrix. The cell in the ith row and jth column represents $p(i|j)$.

	m^0	m^l	m^r
m^0	0.88	0.07	0.05
m^l	0.21	0.79	0
m^r	0.31	0	0.69

the order of the maximum and minimum x-acceleration peaks. The method's thresholds are chosen to maximize the lane-change detection precision and recall using our evaluation data.

5.1.1. Lane change event confusion matrix

The lane change event captures the uncertainty in the sensor measurements and manifests the accuracy of our lane-change detection algorithm. The matrix models the confusion between left-lane-change (m^l), right-lane change (m^r) and no-lane change (m^0). Each cell $cell(i, j)$ of the confusion matrix represents the probability of detecting an event i while actually at event j where $i, j \in \{m^l, m^r, m^0\}$.

We constructed the confusion matrix using our training data with more than 300 traces covering around 260 km. Table 3 shows the confusion matrix used in *LaneQuest*.

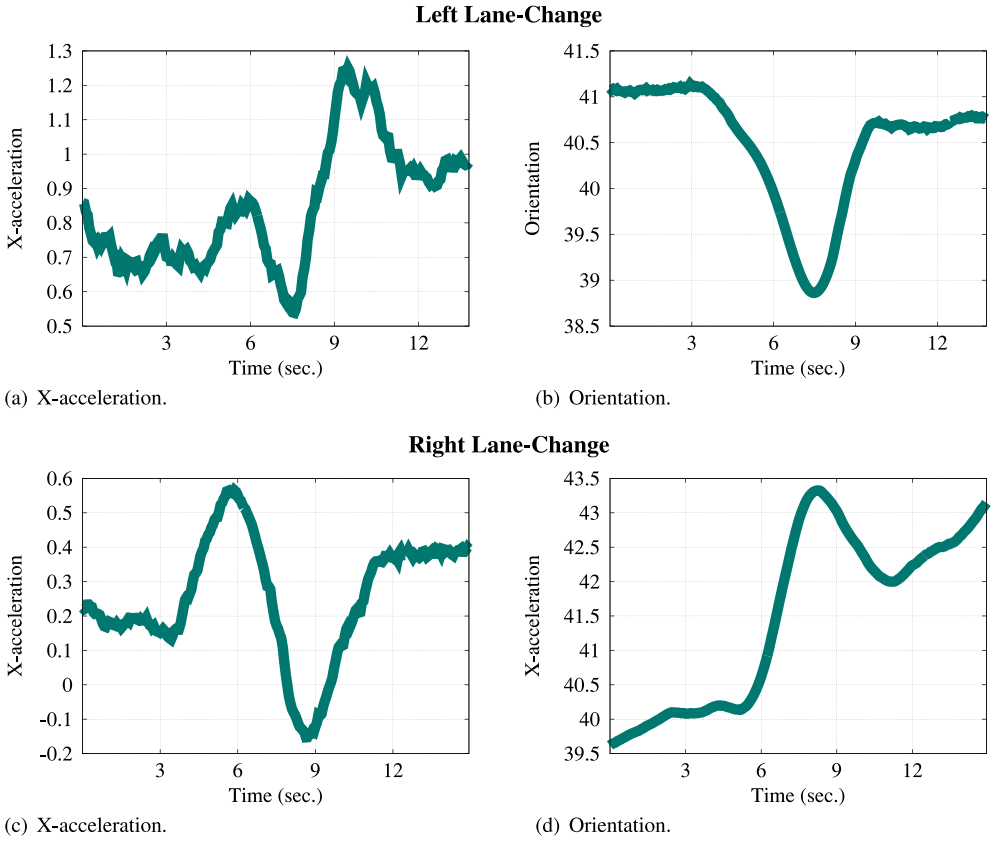


Fig. 6. Left-lane change causes a specific pattern on the *x*-acceleration and just a small peak on the orientation. The *x*-acceleration pattern is reversed when doing right lane-change.

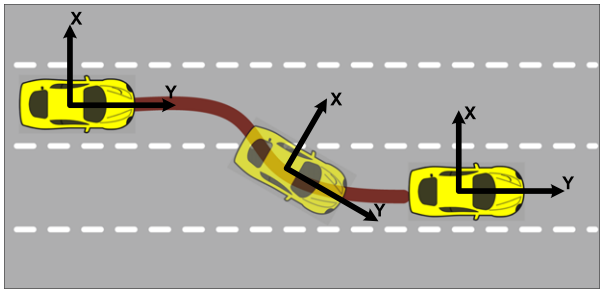


Fig. 7. A car doing a lane change will have to make a small rotation around the *z*-axis, leading to a change in its *x*-axis acceleration.

5.2. Lane anchor detection

5.2.1. Bootstrap lane anchors

LaneQuest defines bootstrap anchors as anchors that have unique sensors signature and a priori known lane distribution. These anchors include turns, merging and exit lanes, and stopping lanes. For the rest of this subsection, we will give details about the anchors detection and lane distribution for each of them. Fig. 8 shows the decision tree used to identify the bootstrap lane-anchors.

Turns:

Turns and u-turns force the car to change its direction by around 90° and 180° respectively, which results in a big variance in the car's orientation along with a change in its final orientation when it ends. This can be captured using the phone's orientation sensor as shown in Fig. 4(b). To further differentiate between right and left turns, the difference between the starting and ending direction can be computed or the *x*-acceleration can be used as it results in patterns similar to the lane-change event (Fig. 6).

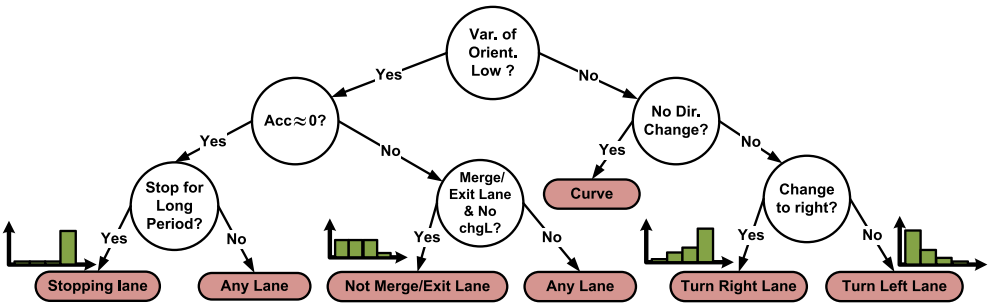


Fig. 8. The decision tree used to identify the lane-known anchors.

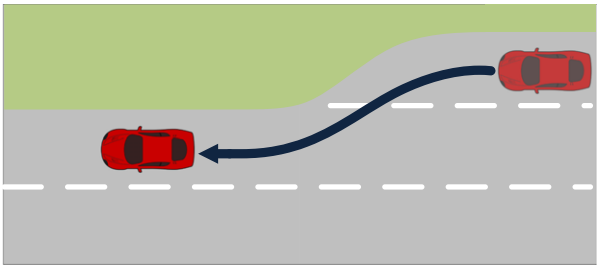


Fig. 9. A lane merge occurs when the road width decreases leading to a decrease in the number of lanes. A car moving at the merged lane will have to change its lane.

Since the driver should make a turn only from the closest lane, the lane distribution for turn anchors is a skewed distribution according to the turn type. This distribution can be used initially and updated dynamically based on the crowd-sensed data as discussed in the next section.

Merge and exit lanes:

A merging lane is used to merge traffic between two roads or to merge traffic when the road width changes (e.g. Fig. 9). Similarly, an exit lane is used to exit a road, e.g. a highway, to another. Usually these lanes have a special extra lane to the main lanes on the road. The location of these lane anchors can be extracted from the digital map and passing by them can be detected based on the car's map-matched location. These lanes are usually the last lanes to the right or left. Therefore, if a car uses an exit or merge lane, its lane distribution will be skewed. Note also that not taking an exit or merge lane indicates that the car is not located in these special lanes. This negative information can be associated with the complement distribution of this type of anchors.

Stopping lanes:

A car may only park in the right-most lane of a driving road. However, traffic signals and road congestion can make a car stop at any lane. To differentiate between parking and the other cases, we use a simple time filter, where parking is detected only if the car stops for more than 3 min.

A parking anchor distribution clusters mainly on the right-most lane only and has small weights for the other lanes.

To estimate the car speed, we use the speed value returned by the location sensor when it is available. Otherwise, we compute it from the consecutive location estimates.

5.2.2. Organic lane anchors

LaneQuest also defines organic anchors which have unique sensors characteristics across the different lanes. However, their lane distribution and road position cannot be predetermined without war-driving. These anchors include curves, tunnels, and potholes. For the rest of this subsection, we will give details about these anchors unique characteristics and how they differ across the different lanes. We leave the details of learning their characteristic in an "organic" way to the next subsection.

Curves:

When a vehicle drives over a curved-road with radius r, the direction of its tangential velocity vector (v) changes as it rotates over the curve. The rate of the direction change is the centripetal acceleration (a), which always points inwards along the radius vector of the circular motion. Without this acceleration, the vehicle would move in a straight line, according to Newton's laws of motion. Based on the circular motion laws [35], the magnitude of the centripetal acceleration (a) is related to the tangential speed (v) and angular velocity (ω) as (Fig. 10):

$$a = \frac{v^2}{r} = \omega^2 r, \tag{16}$$

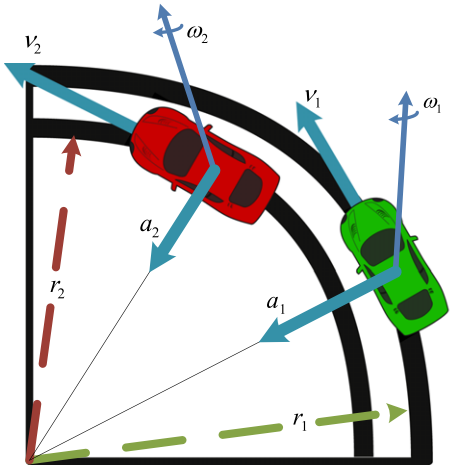


Fig. 10. While moving over a curved road with radius r_i, the magnitude of the centripetal acceleration (a_i) is related to the tangential speed (v_i) and angular velocity (ω_i).

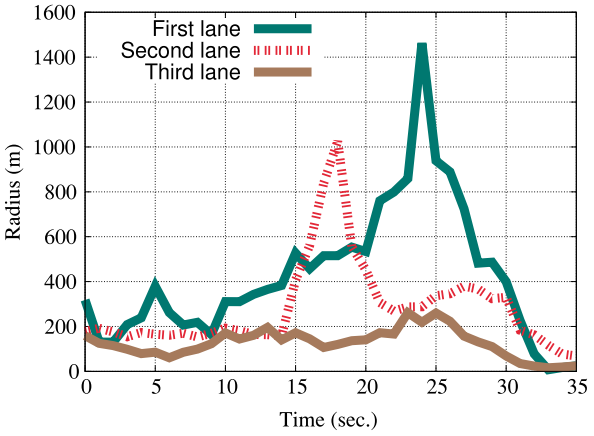


Fig. 11. Estimated radius for a car moving at the different lanes of the same curve.

which can be arranged as:

$$r = \frac{a}{\omega^2} \tag{17}$$

$$r = \frac{v^2}{a}. \tag{18}$$

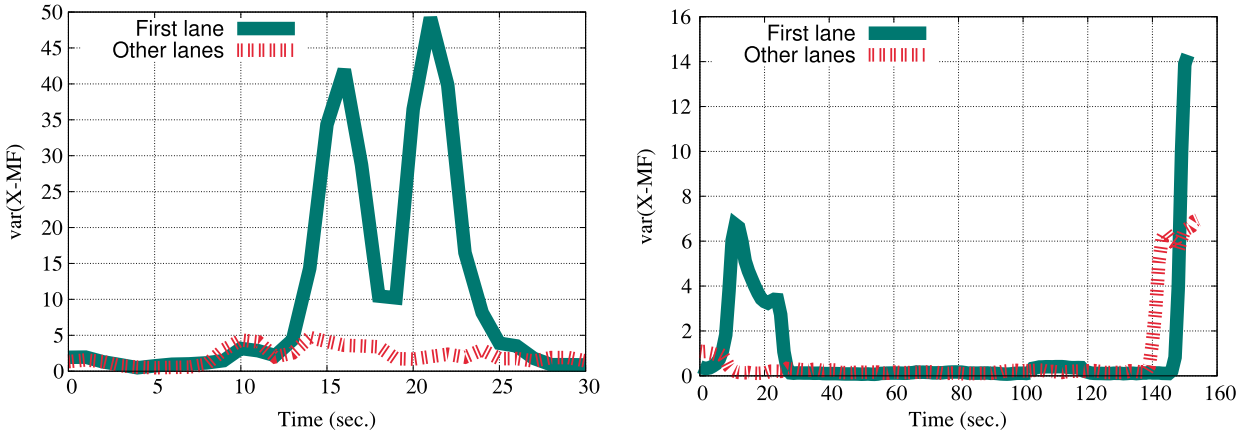
These equations provide two methods for estimating the radius of the lane where the car is driving based on the inertial sensors (angular and linear velocity). Since the gyroscope's measured angular velocity is more accurate in the short term [36], we use Eq. (17) if the phone has a gyroscope sensor. Otherwise, we revert to Eq. (18).

Fig. 11 shows an example of the radius estimated for road curves at the different lanes. We can see a clear distinction between them.

Note that to differentiate between turns and curves, we rely on the change in the user's direction before and after the curve. If the user is taking a turn, the user's direction will have a high change (e.g. 90°). However, if the user is driving over a curve, her direction after finishing the curve will be almost similar to her starting direction [6]. This can be further improved by incorporating the probability of taking a turn or a curve based on the user's location.

Tunnels:

Going inside a tunnel causes a drop in the cellular signals for all the heard cell-towers [6]. This drop can be used to detect the tunnel, but not the specific lane inside the tunnel as it is sensed in all lanes. Studying the effect of moving inside large tunnels with a number of lanes, we noticed a large variance in the ambient magnetic field in the x-direction (perpendicular to the car direction of motion) while the car is going inside the tunnel and going out of the tunnel. This can be explained by the metal and infrastructure (e.g. electricity lines) that exist on the side of the tunnel structure. This high variance decreases as



(a) Estimated curvature radius.

(b) Variance of the estimated radius.

Fig. 12. As the car goes inside and outside the tunnel, it experiences a higher variance in the x-magnetic field. As we move away from the lane closest to the infrastructure, the variance decreases.

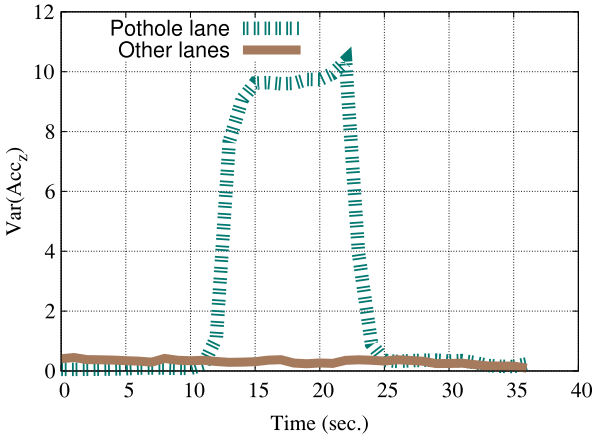


Fig. 13. While moving over a curved road with radius r_i, the magnitude of the centripetal acceleration (a_i) is related to the tangential speed (v_i) and angular velocity (ω_i).

you move away from the tunnel's side where the infrastructure is installed (Fig. 12). This is expected as magnetic interference is known to have an effect on smartphone's magnetometer within small distances only [37].

Potholes and other anomalies:

Anomalies in the road surface such as potholes span only part of the road compared to traffic calming device (e.g. bumps and cat's eyes), which spans the whole road. We identify such anomalies using thresholding on the variance of the z-gravity acceleration as in [24] (Fig. 13). However, this leads to an ambiguity with other traffic calming devices. To resolve this ambiguity, we further use our unsupervised learning approach described in the next section. Typically, a traffic calming device such as a bump will have a uniform distribution over all lanes compared to a pothole that has a narrow distribution.

5.3. Organic lane anchors automatic detection

Organic anchors have known sensors signature but their exact location in the road and their probability distribution across the lanes cannot be predetermined unless a calibration phase across the area of interest is employed. Typically, this imposes an arduous data collection at the different lanes for the entire area. To reduce this overhead, we propose *an unsupervised crowd-sourcing approach* for identifying these lane-anchors profile. Specifically, for each identified road-anchor (e.g. a curve lane), we aim to determine its road location as well as its lane span distribution. Our analysis shows that, in general, our lane-anchors expose different signatures across the road's lanes.

We use **a three-step process** to determine the lane anchor profile (Fig. 14). Without loss of generality, we use the curve lane anchor as an example. *First*, we apply spatial clustering on *all crowd-sourced samples* sent from all the users to the *LaneQuest* server that are detected as curves using their input location. This separates the different curves over the area of interest (i.e. identifying the road anchors). The road location of the road anchor is taken as the centroid of all points

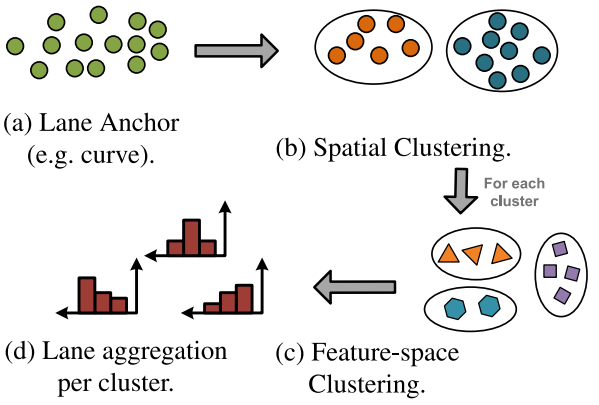


Fig. 14. The three-step unsupervised crowd-sourcing approach used by *LaneQuest* to learn the road location and lane distribution of the organic lane anchors.

within this cluster. *Second*, for each resulting cluster (representing one specific curve), we do a second level clustering of its points based on the lane-discriminating features (the radius in this case as explained in Section 5.2.2) to separate the curve lane-anchors.[2] This helps in determining the lane position for a given lane anchor. *Third*, the curve lane anchor probability distribution ($P(a|\ell)$) is constructed from all the reported car lane beliefs for the points within this last resulting feature-based clustering step. In particular, we take the average of lane beliefs of the different points inside the cluster as:

$$P(a|\ell) = \frac{1}{c}\sum_{i=1}^{c} P(L = \ell|u_i) \tag{19}$$

where c is the number of points inside the cluster and $P(L = \ell|u_i)$ denotes the probability that car u_i was at lane ℓ when it passed by this lane anchor.

Finally, we note that we apply a density-based clustering algorithm (DBSCAN [38]) for the two level clustering algorithms as the number of clusters is not required to be known in advance, the detected clusters can have arbitrary shapes, and outliers can be detected.

6. Discussion

In this section, we discuss different aspects of the *LaneQuest* system.

6.1. Localization source

LaneQuest can be used with any coarse-grained smartphone-based localization system, e.g. GPS, network-based localization systems, or energy-efficient systems like Dejavu [6], as a source for the location-stamped input data (Section 3). Using advanced map matching techniques, e.g. [30,39], *LaneQuest* can further enhance the accuracy of the coarse-grained network-based localization.

6.2. Traffic rules and traffic patterns

Some drivers may violate traffic rules occasionally. For example, a driver can make a right turn from the second right-most lane. Similarly, the driving pattern can change from free-flow to stop-and-go due to congestion at busy roads. For example, in a highly-congested road with stops that exceed our threshold (3 min), a stop-and-go pattern might be confused with stopping lanes. *LaneQuest*, since it uses a probabilistic framework, can incorporate this possibility in its lane distribution by flattening the lane distributions of the different anchors (e.g. by increasing the distribution variance). Also, many lane anchors, e.g. tunnels and potholes, are traffic rules and traffic pattern-independent which can correct and update inaccuracies due to these issues; leading to a high overall lane estimation accuracy as we quantify in Section 7.

6.3. Number of lanes

LaneQuest assumes that the used digital map includes information about the number of lanes for the different roads. Such information is available in a number of the current digital maps and can also be automatically inferred using crowd-sourcing approaches, e.g. [40].

[2] Note that for a given curve, each radius corresponds to a different lane anchor.

Table 4
Summary of the different testbeds used.

Testbed	Distance covered	Speed (km/h)			Number of lanes			Number of roads	Number of lane anchors	
		Min.	Avg.	Max.	Min	Avg.	Max.		Bootstrap	Organic
Alexandria, EG	200	0	40.3	70.0	3	4.6	5	22	359	312
Makkah, KSA	60	0	88.3	106.2	3	3.1	4	8	45	48

6.4. Other smartphone sensors

In this paper, we focused on using inertial sensors to detect the different lane-anchors. Other sensors on the phone, e.g. the camera, can be leveraged to further increase the density of anchors and hence accuracy. Cameras can be used to detect features from the road, e.g. the road lane markers, which can be used as visual anchors. However, careful planning should be employed to balance the energy-accuracy trade-off. For example, the camera can be turned on at low-duty cycle when no other anchors/events are detected for a period of time.

6.5. Special sensors

LaneQuest leverages the smartphone sensors to provide a ubiquitous system where users can use their off-the-shelf smartphone devices to get an accurate lane-estimate. However, the proposed lane estimation algorithm can be easily deployed on a car using external sensors (e.g. by integration with the on-board diagnostic (OBD) unit) if available. This should lead to even better accuracy.

6.6. Vehicle cooperation

Cooperation among vehicles/inter-vehicle communication can be used as possible lane-anchors too. For example, consider that car X (in lane a) has passed by Car Y (in lane b). If we know the relative number of lanes between the two cars, we can update both cars' lane positions accordingly. The concept of device-free sensing has been used to localize multiple entities [41] as well as identify vehicles and their speeds [42,43] through analyzing their effect on the wireless signal strength of standard WiFi devices. This presents a promising technology to estimate the inter-vehicle lane distance.

6.7. Processing location

The *LaneQuest* system can be split into a client–server architecture, where the user's smartphone works as a client that collects sensors data, sends them to the server for processing, and receives the user's lane position from the server. The server applies the probabilistic lane-estimation algorithm and collects the crowd-sourced users' data in a repository to use them in detecting new lane-anchors and identifying their lane position in an unsupervised manner.

Also, we can cache part of the lane anchors database on the client, based on the current estimated location or her trips history, and perform the lane-anchors detection and matching locally on the client. Both techniques have their pros and cons in terms of required resources, latency, and communication cost. The optimal choice depends on the system designer's goals.

7. Evaluation

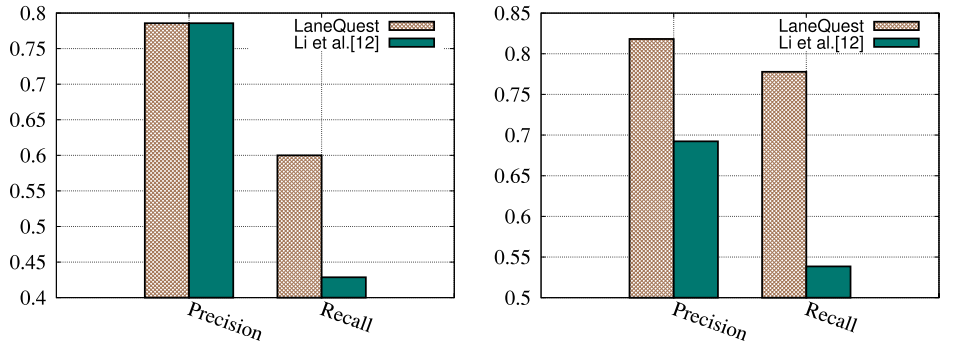
We implemented *LaneQuest* on different Android devices including HTC Nexus One, LG Nexus 4, LG D686, Samsung Galaxy Note, and Samsung Galaxy Nexus. We evaluated the system in the city of Alexandria, Egypt and the city of Mecca, Saudi Arabia covering a combined road length of 200 km in Alexandria and 60 km in Mecca. On average, we had about 4.17 lanes per road in our traces. Table 4 summarizes the testbeds parameters.

To define our lane position ground truth, we developed a simple application to facilitate marking lane positions across the trips and the number of lanes in a particular road. The application has a map with a pointer for the user current position. To set the lane position, the user clicks on the pointer and chooses the number of her current lane. To reduce the error due to manual labeling of the ground truth, the application detects lane changes and prompts the user to confirm the lane change and the new lane position. Without loss of generality, we use GPS as the localization technique throughout this section.

For the rest of this section, we start by evaluating the accuracy of identifying the different events and anchors. Then we show the overall lane estimation accuracy. Finally, we show the energy-consumption overhead of adding *LaneQuest* to different localization systems to estimate the car's lane position.

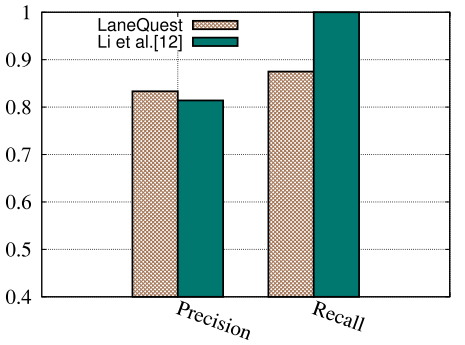
7.1. Lane change detection accuracy

Fig. 15 shows the lane-change detection accuracy using our proposed method as compared to the method proposed in [11]. The figure shows that *LaneQuest*, by taking the variance into account, significantly enhances the accuracy of the lane change effect and reduces the confusion with other events (such as taking a turn or going over a curve).



(a) Left lane-change.

(b) Right lane-change.



(c) No lane-change.

Fig. 15. *LaneQuest* can detect the different lane-change events accurately. Also, the proposed method further improved the lane-change detection accuracy.

Table 5
Confusion matrix for the lane change events and the related anchors (curve). Note that Chg-L/R denotes left/right lane change.

	Turn	ChgL-R	ChgL-L	Curve	Straight
Turn	1	0	0	0	0
ChgL-R	0	0.88	0	0.12	0
ChgL-L	0	0	0.8	0	0.2
Curve	0	0	0	1	0
Straight	0	0	0	0	1

7.2. Motion events and bootstrap anchors detection accuracy

Table 5 provides the confusion matrix for detecting the motion events (i.e. lane change) and the related anchors (i.e. turns and curves). The table shows that we can detect the lane-changes, turns, and curves with high accuracy. This in turn enables high accuracy in lane estimation as we see later. Note that the proposed lane change detection method improved the accuracy of detecting both left and right lane changes by 26.8% and 7.3% respectively as compared to literature.

7.3. Lane anchors detection accuracy

7.3.1. Bootstrap lane anchors detection

Fig. 16 provides the precision and recall for the different bootstrap lane anchors. The figure shows that we can identify the different lane anchors accurately with an average precision and recall of 0.98 and 0.91 respectively.

7.3.2. Organic lane anchors detection

Fig. 17 provides the precision and recall for the different organic lane anchors. Note that curves here reflect the accuracy of detecting the correct lane within the curve as opposed to separating the curve from other events in the confusion matrix. The figure shows that we can identify the different organic lane anchors accurately with an average precision and recall of 0.91 and 0.88 respectively.

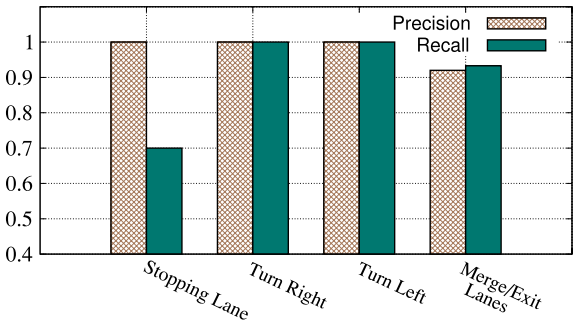


Fig. 16. *LaneQuest* can identify the different **bootstrap lane anchors** with an average precision and recall of **0.98** and **0.91** respectively.

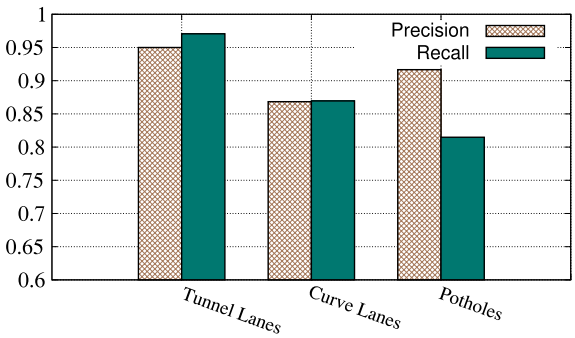


Fig. 17. *LaneQuest* can identify the different **organic lane anchors** with an average precision and recall of **0.91** and **0.88** respectively.

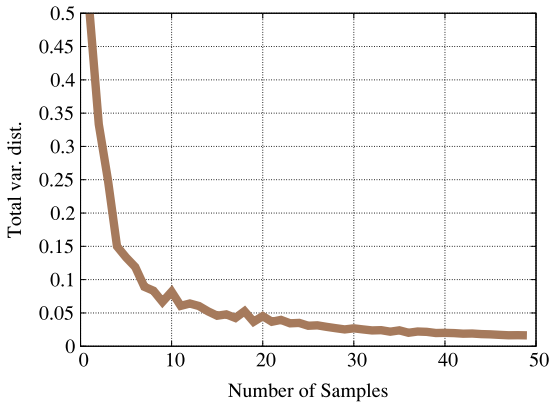


Fig. 18. Effect of number of points within a cluster traces on the accuracy of identifying the lane-anchors.

Fig. 18 shows the effect of the number of points within a cluster of traces on the organic lane-anchors identification accuracy reflected by a zero total variation distance [44] between successive distributions. The figure shows that our two-stage clustering algorithm converges to a stable lane anchor distribution using as few as 20 points per organic anchor. This number is even amortized over the different cars that pass by this specific anchor.

7.4. Lane estimation accuracy

In this section, we evaluate the lane estimation accuracy of *LaneQuest*. We start by evaluating the performance of the *maxBel* and *minErr* methods proposed in Section 4.5. Then, we evaluate the effect of the lane events frequency on *LaneQuest*.

7.4.1. Effect of the lane estimation methods on accuracy

Fig. 19 shows the CDF of the overall lane estimation error (transient+ steady state) for *LaneQuest* (while using *maxBel* and *minErr* methods) as compared to GPS and *LaneQuest*. For GPS, we take the lane estimate as the closest lane to the reported GPS location. Due to the GPS inaccuracy, GPS biases its lane estimate to the rightmost or leftmost lane, leading to a large

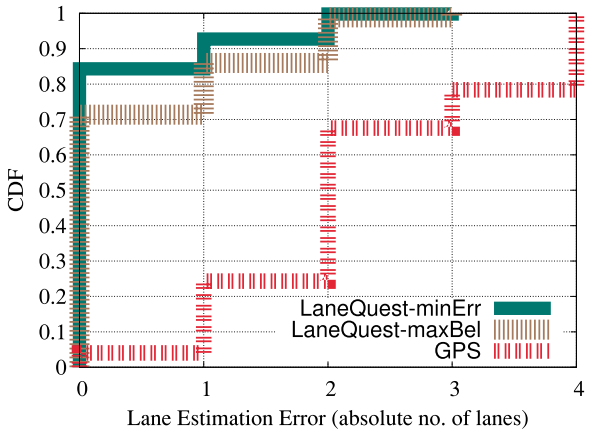


Fig. 19. CDF of the overall lane estimation error for *LaneQuest* when using the *maxBel* and the *minErr* methods as compared to GPS.

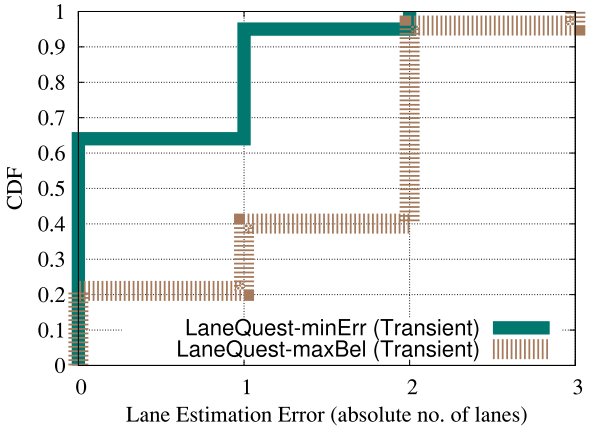


Fig. 20. CDF of lane estimation error **incurred at the beginning**, i.e. transient period, when using the proposed lane estimation technique along with the *maxBel* and the *minErr* methods.

error in lane estimation. On the other hand, *LaneQuest* can identify the car's exact lane more than 84% of the time while using *minErr* method as compared to 70% of the time when using *maxBel* method. Hence, the proposed *minErr* lane estimation technique could improve the lane estimation accuracy more than 20% as compared to choosing the most probable lane (*maxBel* method introduced in [13]).

We note that *LaneQuest* incurs the highest errors at the beginning as it starts from an unknown lane position (uniform distribution). However, our proposed *minErr* lane estimation technique reduces these transient errors significantly as shown in Fig. 20 by more than 200% compared to the *maxBel* method.

7.4.2. Effect of lane event frequency on accuracy

While we expect a user to detect a large number of lane-events (e.g. lane-changes, curves, turns, etc.), typically, we cannot predict how many lane-events the user will encounter during a trip. For this, we study the effect of reducing the frequency of detected lane anchors on accuracy by sub-sampling the actual detected events (Fig. 21). The figure shows that even with a low rate of 10 lane-events detection per hour, *LaneQuest* can still identify the car's exact lane more than 60% of the time.

7.4.3. Effect of outliers on the lane estimation accuracy

Drivers violating traffic rules as well as different traffic patterns can cause false positives and false negatives to our lane anchor detection module. False negatives represent missed error-resetting opportunities while false positives represent incorrectly detected lane anchors. We have discussed the effect of lane anchors frequency, which corresponds to the false negative rate, in Section 7.4.2. Through this section, we discuss the effect of false positives on *LaneQuest* lane estimation accuracy.

Fig. 22 shows that *LaneQuest* could detect the car's exact lane 80% of the time even with an outliers density of 2 per hour. However, the *LaneQuest* system is able to quickly recover from these errors and achieve high accuracy because of (1) flattening the lane distributions of the different anchors and (2) the ample lane-anchors (Section 5).

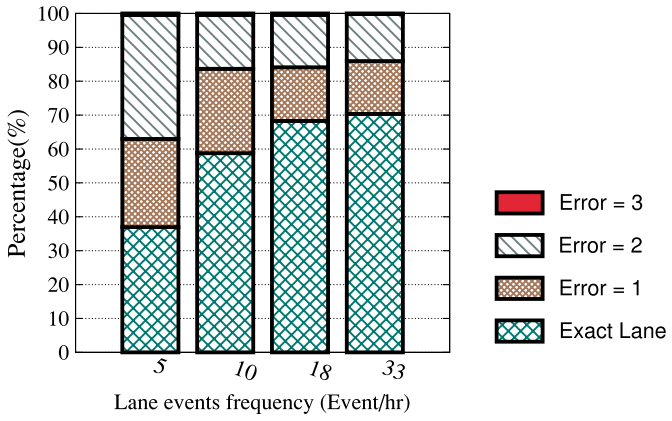


Fig. 21. Effect of events frequency on lane estimation accuracy.

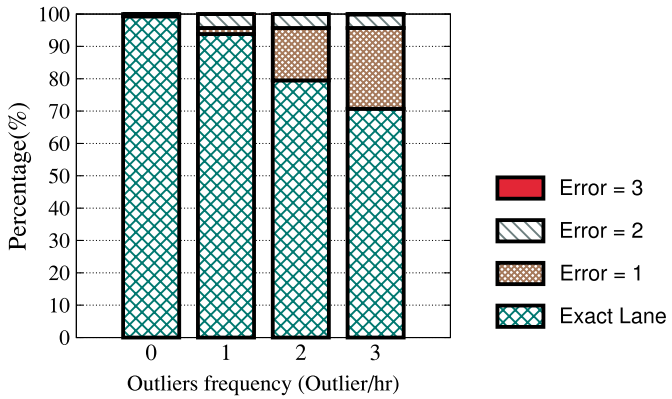


Fig. 22. Effect of the outliers density on *LaneQuest* lane estimation accuracy.

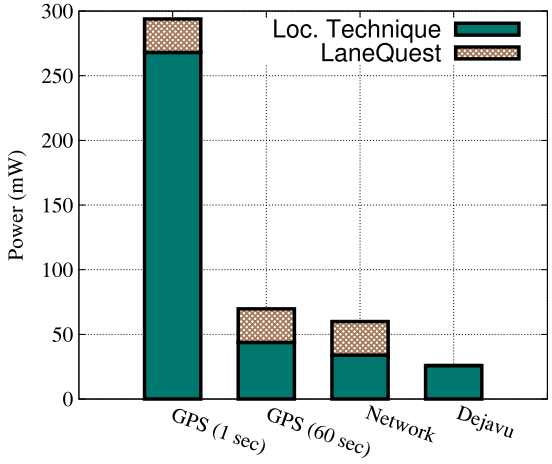


Fig. 23. Power consumption for the different systems when integrated with *LaneQuest*: GPS with sampling rate of 1 sample every second and 1 sample every minute, network-based, and the inertial-sensors-based system (Dejavu [6]).

7.5. Energy overhead

Fig. 23 shows the energy overhead when integrating *LaneQuest* with other localization systems: GPS with sampling rates of 1 sample every second and 1 sample every minute, network-based localization, and the energy efficient inertial-sensors-based localization system (Dejavu [6]). The power consumption was calculated using the PowerTutor profiler [45] and the Android APIs using the HTC Nexus One cell phone. Even though we implemented *LaneQuest* on GPS only, we compare its

energy consumption to other localization systems based on estimating their energy consumption from the sensors they use. The figure shows that *LaneQuest* has a small negligible energy footprint. In addition, when combined with systems that use the inertial sensors for localization, e.g. Dejavu [6], it consumes zero extra energy. This highlights its suitability for use with the energy-constrained mobile devices.

Moreover, we measured the battery discharge rate when using *LaneQuest* and when using the camera through running both the camera and the *LaneQuest* system on fully-charged devices. On average, using the camera decreases the battery life time by more than 96% relative to *LaneQuest*. This emphasizes the advantage of *LaneQuest* as compared to smartphone camera-based systems (e.g. in [12,16]).

8. Conclusion

We presented the *LaneQuest* system for providing a robust accurate estimate of the car's lane position using only energy-efficient inertial sensors available on off-the-shelf smartphones; without any prior assumption on its starting lane position. *LaneQuest* employs a novel Markov-localization based probabilistic framework that fuses knowledge of the vehicle's dynamics with lane-anchors. These anchors are learned through an organic crowd-sourcing approach.

Implementation of *LaneQuest* on a number of Android devices using typical driving traces at different cities shows that it can detect the different lane-level landmarks with an average recall and precision of more than 91%. This enables it to detect the correct lane accurately more than 84% of the time, increasing to 92% of the time to within one lane error. This comes with a low energy profile, allowing it to be implemented on the energy-constrained mobile devices.

Currently, we are extending the system in multiple directions including experimenting with other motion and perception models, extracting more lane-level anchors, using other phone sensors, among others.

Acknowledgments

This work was supported in part by a Google Research Award to E-JUST and in part by the KACST National Science and Technology Plan under grant #11-INF2062-10, and the KACST GIS Technology Innovation Center at Umm Al-Qura University under grant #GISTIC-13-09.

References

[1] S. Hofmann, C. Brenner, Quality assessment of automatically generated feature maps for future driver assistance systems, in: SIGSPATIAL GIS, ACM, 2009.
[2] J. Markoff, Google cars drive themselves, in traffic.
[3] A. de Palma, R. Lindsey, Traffic congestion pricing methodologies and technologies, Transp. Res. C 19 (6) (2011) 1377–1399.
[4] A. Doshi, B. Morris, M.M. Trivedi, On-road prediction of driver's intent with multimodal sensory cues, IEEE Pervasive Comput. 3 (2011) 22–34.
[5] A. Doshi, M.M. Trivedi, Tactical driver behavior prediction and intent inference: A review, in: ITSC, IEEE, 2011.
[6] H. Aly, M. Youssef, Dejavu: an accurate energy-efficient outdoor localization system, in: SIGSPATIAL GIS, ACM, 2013.
[7] D. Bétaille, R. Toledo-Moreo, Creating enhanced maps for lane-level vehicle navigation, IEEE Trans. Intell. Transp. Syst. 11 (4) (2010) 786–798.
[8] R. Toledo-Moreo, D. Bétaille, F. Peyret, Lane-level integrity provision for navigation and map matching with gnss, dead reckoning, and enhanced maps, IEEE Trans. Intell. Transp. Syst. 11 (1) (2010) 100–112.
[9] R. Toledo-Moreo, D. Bétaille, F. Peyret, J. Laneurit, Fusing gnss, dead-reckoning, and enhanced maps for road vehicle lane-level navigation, IEEE J. Sel. Top. Sign. Proces. 3 (5) (2009) 798–809.
[10] Z. Tao, P. Bonnifait, V. Fremont, J. Ibañez-Guzman, et al. Lane marking aided vehicle localization, in: IEEE ITSC 2013.
[11] D. Li, T. Bansal, Z. Lu, P. Sinha, Marvel: multiple antenna based relative vehicle localizer, in: Mobicom, ACM, 2012.
[12] F. Ren, J. Huang, M. Terauchi, R. Jiang, R. Klette, Lane Detection on the iPhone, Springer, 2010.
[13] H. Aly, A. Basalamah, M. Youssef, LaneQuest: An accurate and energy-efficient lane detection system, in: PerCom, IEEE, 2015.
[14] T.-S. Dao, K.Y.K. Leung, C.M. Clark, J.P. Huissoon, Markov-based lane positioning using intervehicle communication, IEEE Trans. Intell. Transp. Syst. 8 (4) (2007) 641–650.
[15] A. Selloum, D. Betaille, E. Le Carpentier, F. Peyret, Lane level positioning using particle filtering, in: ITSC'09, IEEE, 2009.
[16] P. Chanawangsa, C.W. Chen, A new smartphone lane detection system: realizing true potential of multi-core mobile devices, in: Proceedings of the 4th Workshop on Mobile Video, ACM, 2012, pp. 19–24.
[17] Q. Li, L. Chen, M. Li, S.-L. Shaw, A. Nuchter, A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios, IEEE Trans. Veh. Technol. 63 (2) (2014) 540–555.
[18] M. Fazeen, B. Gozick, R. Dantu, M. Bhukhiya, M.C. González, Safe driving using mobile phones, IEEE Trans. Intell. Transp. Syst. 13 (3) (2012) 1462–1468.
[19] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, R.P. Martin, Sensing vehicle dynamics for determining driver phone use, in: MobiSys, ACM, 2013.
[20] P. Singh, N. Juneja, S. Kapoor, Using mobile phone sensors to detect driving behavior, in: Proc. of the 3rd ACM Symp. on Computing for Development, ACM, 2013.
[21] H. Aly, A. Basalamah, M. Youssef, Map++: A crowd-sensing system for automatic map semantics identification, in: SECON, IEEE, 2014.
[22] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, H. Balakrishnan, The pothole patrol: using a mobile sensor network for road surface monitoring, in: MobiSys, ACM, 2008.
[23] P. Mohan, V.N. Padmanabhan, R. Ramjee, Nericell: rich monitoring of road and traffic conditions using mobile smartphones, in: SenSys, ACM, 2008.
[24] A. Mednis, G. Strazdins, R. Zviedris, G. Kanonirs, L. Selavo, Real time pothole detection using android smartphones with accelerometers, in: DCOSS, IEEE, 2011.
[25] W.S. Cleveland, S.J. Devlin, Locally weighted regression: An approach to regression analysis by local fitting, J. Amer. Statist. Assoc. 83 (403) (1988) 596–610.
[26] N. Mohssen, R. Momtaz, H. Aly, M. Youssef, It's the human that matters: Accurate user orientation estimation for mobile computing applications, in: Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MOBIQUITOUS'14, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2014, pp. 70–79.
[27] M. Ibrahim, M. Youssef, Cellsense: An accurate energy-efficient GSM positioning system, IEEE TVT.
[28] M. Ibrahim, M. Youssef, A hidden Markov model for localization using low-end GSM cell phones, in: ICC, IEEE, 2011.

[29] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, et al., Place lab: Device positioning using radio beacons in the wild, in: Pervasive Computing, Springer, 2005.
[30] R. Mohamed, H. Aly, M. Youssef, Accurate and efficient map matching for challenging environments, in: SIGSPATIAL, ACM, 2014.
[31] S.J. Russell, P. Norvig, J.F. Canny, J.M. Malik, D.D. Edwards, Artificial Intelligence: A Modern Approach, Prentice Hall, Englewood Cliffs, 1995.
[32] W. Burgard, S. Thrun, Markov localization for mobile robots in dynamic environments, J. Artificial Intelligence Res. (1999) 391–427.
[33] A.H. Ang, W.H. Tang, Probability concepts in engineering, Planning.
[34] P. Newson, J. Krumm, Hidden Markov map matching through noise and sparseness, in: SIGSPATIAL GIS, ACM, 2009.
[35] R. Serway, J. Jewett, Physics for Scientists and Engineers, Cengage Learning, 2013.
[36] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, R.R. Choudhury, No need to war-drive: Unsupervised indoor localization, in: MobiSys'12, 2012.
[37] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, M. Wiseman, Indoor location sensing using geo-magnetism, in: MobiSys, ACM, 2011.
[38] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: KDD, Vol. 96, 1996.
[39] H. Aly, M. Youssef, semMatch: Road semantics-based accurate map matching for challenging positioning data, in: Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2015.
[40] Y. Chen, J. Krumm, Probabilistic modeling of traffic lanes from GPS traces, in: SIGSPATIAL, ACM, 2010.
[41] I. Sabek, M. Youssef, A.V. Vasilakos, ACE: An accurate and efficient multi-entity device-free wlan localization system, IEEE Trans. Mob. Comput. 14 (2) (2015) 261–273.
[42] A. Al-Husseiny, M. Youssef, Rf-based traffic detection and identification, in: Vehicular Technology Conference (VTC Fall), 2012 IEEE, IEEE, 2012, pp. 1–5.
[43] N. Kassem, A.E. Kosba, M. Youssef, Rf-based vehicle detection and speed estimation, in: 75th Vehicular Technology Conference (VTC Spring), IEEE, 2012, pp. 1–5.
[44] D.A. Levin, Y. Peres, E.L. Wilmer, Markov Chains and Mixing Times, American Mathematical Soc., 2009.
[45] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R.P. Dick, Z.M. Mao, L. Yang, Accurate online power estimation and automatic battery behavior based power model generation for smartphones, in: CODES/ISSS, ACM, 2010, pp. 105–114.