IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 22, 2019, accepted January 7, 2020, date of publication February 5, 2020, date of current version March 19, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2971849

Givs: Fine-Grained Gesture Control for Mobile Devices in Driving Environments

LANDU JIANG[1], MINGYUAN XIA[1], XUE LIU[1], (Fellow, IEEE), AND FAN BAI[2]
[1]School of Computer Science, McGill University, Montreal, Quebec H3A 0G4, Canada
[2]General Motors Global R&D, Detroit, MI 48243, USA

Corresponding authors: Xue Liu (xueliu@cs.mcgill.ca) and Fan Bai (fan.bai@gm.com)

ABSTRACT New media and communication technologies like mobile devices are nowadays widely used everywhere for providing rich functionalities and highly personalized services. However, using such a device in a driving environment is still very inconvenient and unsafe to be controlled by the driver. The touchscreen operations are one major obstacle since multi-touchscreen is optimized for hand-held usage scenarios. To overcome this limitation, we propose to replace some most used touch operations with gesture controls for mobile devices in a driving environment. Gesture control is simple, more flexible and requires less eye focus, which makes it more suitable for in-vehicle usages. In this paper, we design Givs, a fully functional gesture control system for mobile devices in a driving environment. Givs leverages the latest motion sensing technology to enable ubiquitous and driving-friendly gestures. Compared to other off-the-shelf gesture recognition solutions, Givs is optimized for in-vehicle use cases and is designed to overcome various limitations caused by real driving conditions, including bumpy road conditions, significant noise introduced by car vibration and technical limitations of motion sensors. Our extensive in-vehicle tests and participant experience experiments demonstrate that Givs well assists users in accomplishing various types of tasks and support human-machine interaction in driving environments such as personal vehicle and public transport, with high accuracy and fast responsiveness, while promoting drivnig convenience and safety.

INDEX TERMS Human-machine interaction, smart sensing, mobile computing, driving safety.

I. INTRODUCTION

Mobile devices such as smartphones and tablets have gained great popularity over the past few years. These devices and lots of available applications (apps for short) offer personalized services which provide great convenience to our daily lives. Consequently, these devices now become the most personal devices and accompany users almost everywhere. However, using a hand-held device in a driving environment is still very inconvenient and unsafe. Existing interaction methods, such as touch screen, tactile buttons and speech recognition provide limited functionalities or incur notable driver distraction and safety hazards. Many countries including US actively prohibit drivers from using hand-held devices in a driving environment [5]. To enable convenient and safe use of gestures, various humancomputer interaction studies [8], [15], [27], [33] have been conducted to reduce distraction and understand the effectiveness of different interaction methods. These studies outline that gesture interfaces are more intuitive to users and can effectively reduce visual demands as compared to conventional touch and tactile based interfaces.

A practical in-vehicle gesture system is, however, constrained by various safety requirements and complicated by random road conditions and limitations of motion tracking hardware and software (e.g. gesture recognizers). To understand these challenges, we conduct series of in-vehicle driving experiments and identify the following key design considerations for an effective in-vehicle gesture control system:

1) Physical Constraints: Vehicles could encounter various lighting conditions and users have limited space to perform gestures. These require a gesture sensor to deliver accurate positioning regardless of environmental light intensity and interference of visible light.
2) Environmental Noises: Car vibration, biological hand shaking and sensing errors could significantly degrade the performance of gesture recognizers and these noises must be the first-priority design consideration for recognition software.

The associate editor coordinating the review of this manuscript and approving it for publication was Yongtao Hao.

3) Delay Sensitivity: Gesture recognizers as well as sensing hardware impose latencies. When gestures are used to control delay-sensitive devices, such as a mobile device, delay and jitter must be carefully managed.

In this paper, we focus our research effort on developing and optimizing gesture control mechanisms to address above-mentioned challenges. We propose Givs, the gesture-in-vehicle system enable convenient and safe use of gestures in a driving environment. The gesture control mechanisms proposed in this paper, to our best knowledge, have not been developed or offered by mainstream in-car phone project solutions (i.e. CarPlay [3], Android Auto [2] and MirrorLink [24]). To meet physical constraints (lighting and interaction area), Givs leverages infrared camera based motion sensors that could produce fine-grained hand model within a reasonable 3D space.

Givs redesigns gesture recognizers to overcome driving noises and to meet delay requirements. In particular, Givs comprises a novel differential method that leverages the relative movement between different fingers to null out the impact of environmental noises. Givs also utilizes Holt's linear exponential smoothing techniques for forecasting future hand position in case of delay jitters.

To enhance Givs's usability on complicated real road conditions, Givs also incorporates a bump/pothole detector to make gesture control further safer and more adaptive. Specially, when the car experiences a complicated road condition with lots of bumps and potholes, Givs will temporarily discard gesture inputs and warn the user to keep both hands on the control wheel if he/she is still performing gestures.With this, Givs can promote proper and safe use of gestures while reduce recognition errors at the same time.

Our contribution is three-fold:

1) First, we conduct a comprehensive study of motion sensing in real driving environments. This study outlines key design considerations of an in-vehicle gesture system with regards to sensing hardware, recognition software, delay constraints and road condition impacts.
2) Second, we propose Givs that enables gesture control of mobile devices in driving environments. We develop effective recognition algorithms and delay control methods to address practical challenges we identified.
3) Third, we demonstrate the usability of Givs with extensive user experience study. This study outlines the user's satisfaction with a gesture control system and provides important guidelines for effective gesture design and use in a driving environment.

The paper continues as follows. Section 2 presents the measurement study of motion sensing under various driving conditions. Section 3 elaborates the design of Givs, including its gestures and associated recognition algorithms. Section 4 evaluates Givs with extensive real driving tests. Section 5 summarizes our related work and Section 6 concludes the paper.

II. MOTION SENSING UNDER REAL DRIVING ENVIRONMENTS

In this section, we provide the essential background about motion sensing and present a measurement methodology to assess various design considerations for a gesture control system in real driving environments.

A. MOTION SENSING BACKGROUND

Motion sensing aims to track the movement of physical objects (commonly human body or parts). Movement trails can be used further to recognize gestures and enable more sophisticated functionalities. In driving environments, hand gestures are the most feasible interaction method and thus are the focus of this work.

Motions of the hands or fingers are captured by motion sensors, and touchscreen [15], [27], [33] may be considered as a simple motion sensor that functions in a 2D surface only. By using depth/infrared cameras [12], [22] or WiFi signals [1], [20], [35] modern motion sensors are able to track objects in a 3D space. In this paper, we focus on 3D hand tracking as it competes conventional touchscreen in terms of accuracy while enabling greater interaction flexibility. At each moment, the motion sensor reports a recognized hand model, specifying the position of different parts. Over time, a hand movement trail will be captured and reported. On top of motion sensing hardware, various gesture recognizers [30], [40] aim to recognize patterns (gestures) from hand trails. Each of these recognizers has distinct accuracy, recognition speed and prerequisites (e.g. training). We discover that environmental noises in vehicle could significantly degrade the performance of these existing recognizers, where noise cancellation becomes a key design consideration. We will elaborate these challenges in the following parts of this section. Finally, on top of gesture recognizers, a control module is responsible for translating gestures into commands to control another system (e.g. car infotainment systems, connected mobile devices, etc). At this layer, delay becomes the major concern as some controlling commands are delay-sensitive. Such delay may come from either sensing hardware or recognition algorithms, and will be carefully analyzed in this work.

B. MEASUREMENT METHODOLOGY

In this project, we consider a use case that has not yet been systematically studied before. Users perform hand gestures in a 3D space and a system captures hand movement with a motion sensor, recognizes gestures and generates commands to control connected mobile devices.We propose this measurement methodology to assess various design considerations of such a system, mostly focusing on how driving environments affect different building blocks of the system.

1) PHYSICAL CONSTRAINTS AND HARDWARE SELECTION

Driving environments impose several challenges for motion sensing hardware. Vehicle users have limited space to interact

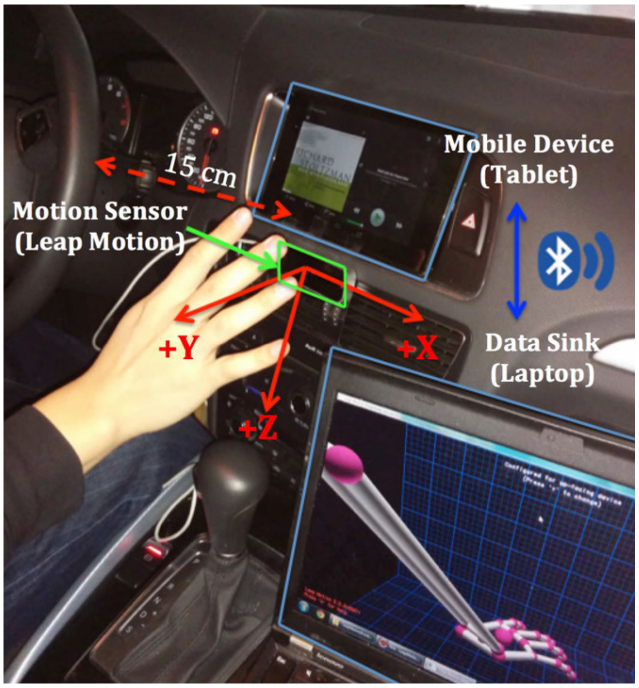


FIGURE 1. The in-vehicle experiment setup. A laptop is used for gesture recognition, visualization and controlling. The Leap Motion sensor is connected to the laptop via USB 3.0, reporting hand positions. The mobile device (tablet) is connected to the laptop via Bluetooth and being controlled.

with the sensor and vehicles could encounter different lighting conditions. These physical constraints require sensors to work under different light intensities and accurately track hand movement within a small area. Both 3D tracking devices (WiFi and infrared camera) can adapt to changing lighting conditions while camera based hardware offers higher tracking accuracy (within millimeter error range) and is highly commoditized. In this study, our prototype system leverages the Leap Motion sensor, a commodity infrared camera based motion sensor for 3D object tracking.

2) EXPERIMENT SETUP

Figure 1 shows our in-vehicle experiment setup. We use a Google Nexus 7 Tablet as the target mobile device to be controlled. We do not enforce the position of the mobile device. Following the current practice, the device could be mounted with a car mount holder or screen cast to the dashboard screen. In our experiment, we emulate the later case where the tablet is tied in front of the car dashboard screen. The Leap Motion sensor is mounted onto the frame below the tablet and above the air conditioner. Currently it is steadily mounted but in the future we could imagine it to be built into the frame. We use a laptop computer as the data sink, which is connected with both the sensor (via USB 3.0) and the device (via bluetooth). In the future, the functionality we implemented on the laptop will be part of the car system.

The Leap Motion sensor continuously reports its readings (positions of 19 bones in a hand and matching confidence). The coordinate system used by Leap is marked in the figure. The matching confidence is a value in the range of 0 to 1 with a higher value indicating higher confidence of the hand/bone positions. The laptop visualizes the hand, recognizes gestures and generates corresponding commands to control the tablet. The tablet reports its accelerometer readings back to the laptop.

3) MEASUREMENT TARGETS

Our measurement study focuses on factors that affect the performance of the gesture control system. It aims to answer the following questions: 1) what is the effective interaction area of sensing hardware and whether it is suitable for in-vehicle usage; 2) what are the environmental noises that could affect gesture recognition accuracy; 3) what are the delay composition of such a system; 4) how special road conditions impact such a system.

C. PHYSICAL CONSTRAINTS

This experiment aims to outline if a given motion sensor (Leap Motion in our case) can satisfy the physical constraints to be used in vehicles. Here we show our methodology and only present our results with the Leap Motion sensor. The same methodology also applies to other motion sensors.

1) EFFECTIVE INTERACTION AREA

We intend to identify an area where the motion sensor can most effectively position the hand. To achieve this, we remain seated with safe beat tied and move hands ergonomically. During the 30-minute experiment, we record more than 76,000 frames from Leap Motion, while each frame reports the position of the hand and its bones with a confidence score. For other sensors, similar scores could also serve for this purpose. We use this score as the primary criteria and reveal the interaction area where the sensor can report with high confidence. We observe that the Y coordinate (palm-to-sensor distance) is fairly constant for vehicle users so we project all points to the X-Z plane. Figure 2 plots the heat map of the confidence in the X-Z plane. A solid color indicates a higher confidence and thus higher sensing accuracy. As shown, Leap Motion naturally senses more accurately when the hand is nearby since far-away objects look smaller to the camera. Another observation is that the sensor is roughly equally sensitive on the "+X" and "−X" directions. But it can accurately sense more "+Z" area than "−Z" direction. This is because the sensor needs a clear view of fingers to deliver high matching confidence. When moving in front of the sensor (-Z), the fingers are more far from the sensor than the palm and thus the sensor confidence is lower.

2) TEMPERATURE AND LIGHTING IMPACTS

We alter temperature and lighting conditions within the vehicle and measure their impacts. Specially, we test seven lighting conditions, including sunny day, cloudy day, rainy day, totally dark, night with on-board lighting on. For temperatures, we adjust air conditioning for a range of $15°C$ to $25°C$ (LeapMotion claims an operating temperature from $0°C$ to $45°C$). We redo the interaction area measurements

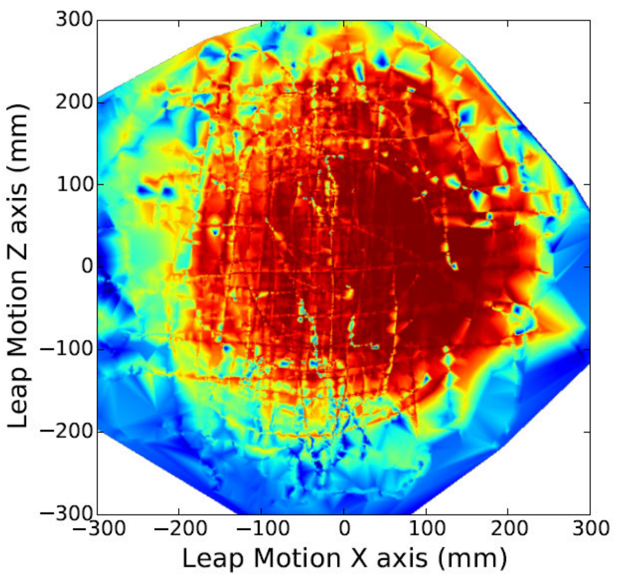


FIGURE 2. The sensing confidence in the X-Z plane. Warmer color (red) denotes higher confidence.

with different settings and manually validate if the effective interaction area shrink or change. Our results show that the Leap Motion sensor, by using two infrared cameras, are unaffected by lighting and temperature conditions in normal vehicle use cases.

D. ENVIRONMENTAL NOISES

Drifting is a very common problem of motion sensing. It denotes the phenomenon that when the user's hand stays still, the sensed position of a particular reference point on the hand varies over time. We consider the drifting noise as the accumulation of the sensor errors, biological hand shaking and car vibration in a driving environment. We include biological shaking in our drift noise because this shaking is inevitable in almost all hand-based control systems. As previous study [12] has revealed, the sensor error is within 0.5mm for all three dimensions. We perform two extra experiments to characterize biological hand shaking and car vibration.

1) BIOLOGICAL HAND SHAKING

We measure the drifting noises in an office environment with the sensor firmly tied on a table. During these experiments, participants intentionally keep the hand still. The distance between the hand and the sensor is 15cm, measured as the distance between the wheel and the dashboard of our car, also shown in Figure 1. The drifting noise measured in this experiment only contains sensing errors and biological hand shaking. From the measured positions, we find that along X and Z axes, human hands tend to have a random drifting noise of 2mm (including the sensor error).

2) CAR VIBRATION

We measure the drifting nosies under various driving environments to characterize the impact of car vibration. Our measurement shows that the car vibration combined with the biological shaking cause a 6mm drifting noises along both axes.

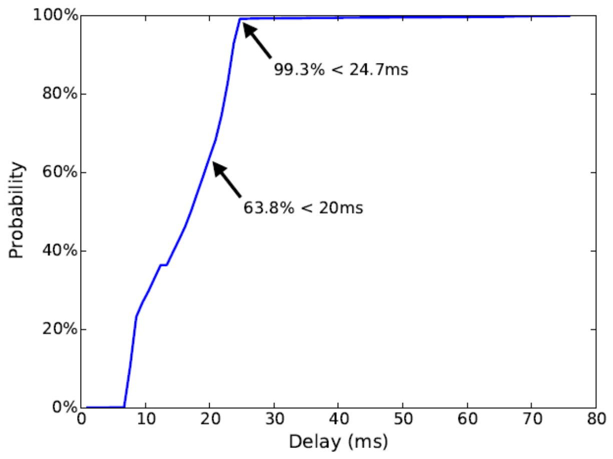


FIGURE 3. Cumulative distribution function of frame delays.

E. DELAY CHARACTERISTICS

Motion sensors repeatedly report hand states in a frame by frame basis. The time interval between two consecutive frames affects the responsiveness of gesture recognizers and in turns the controlling delay. Thus we measure and plot the frame delays to understanding the sampling characteristics of motion sensing. For the Leap Motion sensor, the cumulative delay distribution is shown in Figure 3.We observe that most frame delays ($> 99\%$) fall between 7ms to 25ms, with some outliers that could reach up to 80ms.

This delay variation motivates some key design considerations for Givs. For any delay-sensitive control system that takes Leap Motion as the input source, the delay characteristics of the sensor impact the possible control period. According to Figure 3, the effective control period can be any number that is greater than 80ms (100% delays are within 80ms). For a control system that requires a control period lower than 80ms, delay management will become a crucial design requirement. For example, if the system requires a control period to 25ms, it must tolerate 5% of delays falling between 25ms to 80ms.

F. SUMMARY

Table 1 summarizes the findings from our measurement study, including the effective sensing area, environmental noises and frame delay characteristics. These findings serve as important design considerations for our gesture control system. Though these results are specific to Leap Motion, our measurement methodology is not limited to a specific sensor. For any other motion sensor, the same procedures shall be performed to test the its suitability for gesture control systems and produce key parameters for modules of such systems.

TABLE 1. The summary of in-vehicle measurement study.

Parameter	Value	Description
Effective sensing area		
s_x^+	200mm	The +X limit
s_x^-	200mm	The -X limit
s_z^+	200mm	The +Z limit
s_z^-	100mm	The -Z limit
Environmental noise (in all three dimensions)		
δ_s	0.5mm	Sensor error
δ_b	2mm	Biological shaking
δ_v	6mm	Car vibration
Frame delay		
τ_{max}	80ms	The maximum frame delay
τ_{99}	25ms	The 99th percentile frame delay
$\bar{\tau}$	17.3ms	The average frame delay
Temperature and lighting conditions have no major impacts on sensing performance		

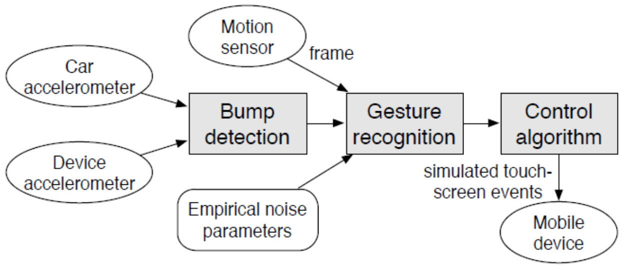


FIGURE 4. The architecture of Givs.

III. GIVS DESIGN

Givs is designed to retrofit the interaction methods for mobile devices in driving environments. It addresses three major challenges. First, proposed gestures should be easily performed by vehicle users in a driving environment, i.e., driving-friendly gestures. Second, gesture recognition algorithms should overcome various noises and sensing limitations in driving environments and recognize gestures with high accuracy. Third, Givs must emit timely control signals to the mobile device since controlling such a device is commonly delay-sensitive.

A. SYSTEM OVERVIEW

Figure 4 depicts the architecture of Givs, which has three major components shown in shaded boxes. The bump detection identifies bumps to detect rough and bumpy roads. The gesture recognition algorithms receive frame updates from the motion sensor and recognize gestures. We redesign gesture recognizers to tolerate noises in driving environments. The recognized gestures will be translated to commands for the mobile device. Two types of gestures are supported currently. One type is a complete sequence of hand movement, denoting one action (e.g pressing the volume button) to the mobile device. Gestures of this type could be perform without eye focus. The other type continuously track hand movement and acts as a virtual mouse cursor on the mobile device screen to the user. We design a control algorithm to tackle delay jitters and produce smooth command flow to the device. Overall, the control algorithm ensures that the whole system can seamlessly transform gestures into continuously and delay-sensitive control events.

B. GESTURE DESIGN AND RECOGNITION ALGORITHMS

Our gesture design aims to propose gestures that can aid vehicle users to easily control mobile devices. At the mean time, these gestures must be effectively recognized, regardless of various environmental noises and interference.

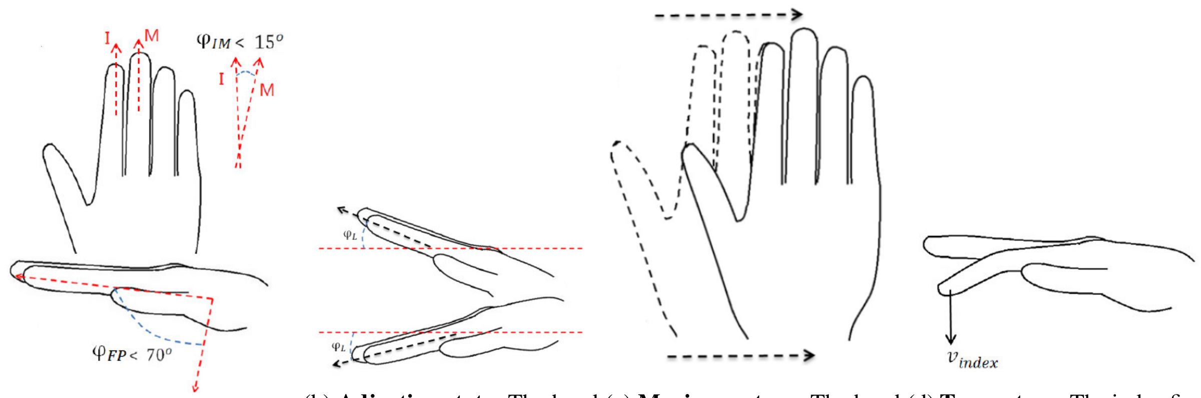
1) HAND STATE TRACKING

a: ACTIVATING STATE

Since the motion sensor continuously senses vehicle user's hand, we need a gesture to distinguish between relaxed state with no user input and activated state with the intent to control the device. A cursor appearing on the device screen indicates the hand state. Under the activated state, the device acts as if a finger tip presses on the cursor. We define the activated state as the situation that the user performs a flat hand with four fingers together and palm facing the sensor, as shown in Figure 5a. We choose this gesture for two reasons. First, this gesture is natural to human beings and is a common gestures (e.g. for diving communication [7] and in sign languages [32]). Second, when the hand is in this state, all bones are approximately aligned to a surface, serving as the reference plane for other gestures. We use two parameters to identify such a gesture. The first parameter is the angle between the palm normal and finger pointing direction, denoted as φ_{FP}. The other parameter is the angle between the pointing direction of the index finger and the middle finger, denoted as φ_{IM}, which indicates whether these two fingers are together. To obtain the thresholds that distinguish activated gesture, we measure the two angles with our ten participants performing relaxed hand gesture and the activating gesture. According to our measurement, we find that the activating gesture is formed when the user has an $\varphi_{FP} > 70°$ and $\varphi_{IM} < 10°$. Our recognition algorithm constantly computes these two angles from the frame reported by the sensor and checks this condition to detect if the hand is activating.

b: ADJUSTING STATE

We define an adjusting gesture to support volume adjustment or zooming operations. This gesture starts from the activating state (flat hand) and requires the hand to lean forwards or backwards. If the hand has a leaning angle beyond a designated threshold for a short while, then Givs interprets this gesture as an adjusting gesture. Otherwise, the adjusting gesture is stopped immediately. The adjusting command can be commonly interpret as a command that changes a particular continuous variable of the target control system. For example, the adjusting gesture can be used to increase or decrease the volume for a music playback application (emulating clicks on volume adjustment keys on the sides of the hand-held devices). Also, the adjusting gesture can serve



(a) **Activating state:** Flat hand with four fingers together. (b) **Adjusting state:** The hand leans forwards/downwards over a designated angle. (c) **Moving gesture:** The hand moves around in activated state. (d) **Tap gesture:** The index finger moves down and restores to almost the original position.

FIGURE 5. Four gestures supported in Givs. We assume all gestures are performed when the palm is facing the motion sensor.

as a zooming command for a map navigation application or an image viewer. The prerequisite of activating state (flat hand) greatly facilitates the detection of adjusting gestures. When the flat hand is maintain, all bones align nearly in the same surface and the leaning angle can be easily computed. In practice, we require users to have a leaning angle over 30° and keep the gesture for at least half second to indicate an adjusting gesture. These two design parameters are subject to user preference but our experiments indicate that these the designated values can greatly eliminate user's accidental operations.

2) HAND POSITION TRACKING

Givs emulates a virtual cursor on the mobile device screen to track the hand movement. Thus hand position tracking is the foundation of our system. Meanwhile, drag-anddrop is a common command for mobile devices, which is commonly used to navigate the map area, swipe between tabs, etc. These application scenarios require accurate and continuous hand position tracking. Givs constantly tracks the palm center and updates the virtual cursor position accordingly. When the hand changes from related to activated state, Givs interprets it as a dragging gesture. We observe the following technical challenges for hand tracking:

1) When the car encounters bumps, the hand is in unstable states and its tracking position will observe large unintended and unpredictable turbulence.
2) Some frames reported by the motion sensor has large delay jitters. Thus the position update is not uniform in the time domain.
3) We observe car and biological drifting that could lead to a drifting cursor.

To overcome these challenges, Givs proposes to use smoothing algorithms to process raw palm position data from the frames reported by the motion sensor. In particular, when a bump is detected, Givs discards the raw data from the sensor and uses the smoothing algorithm to forecast several positions based on previous movement trends. When Givs encounters delay jitters, it will also forecast positions to ensure that position updates come within a hard deadline.

a: HOLT'S LINEAR EXPONENTIAL SMOOTHING

We observe that hands have notable linear movement trends within a small time period. To capture this trend and use it to forecast future positions, we leverage exponential smoothing that considers all previous frames and models trending parameters.

In particular, we employ Holt's linear trend method [34] to effectively forecast time-series data with trends. The estimate of h-th future frame at given time t is derived from two variables l_t (level) and b_t (slope) with the following equation:

$$\widehat{x}_{t+h|t} = l_t + hb_t, \tag{1}$$

In Eq 4, l_t denotes an estimate of the level of the series at time t, which is computed as:

$$l_t = \alpha x_t + (1-\alpha)(l_{t-1} + b_{t-1}), \tag{2}$$

l_t can be interpreted as a weighted average of the observed data point x_t and the one-step-ahead forecast for time t (i.e., $x_{t|t-1}$). bt denotes an estimate of the trend of the series, which is computed as:

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}, \tag{3}$$

b_t is essentially the weighted average of the last trend estimate b_{t-1} and the difference between next and last level (l_t - l_{t-1}).

The two smoothing parameters α and β control whether the forecaster relies more on historical observations or the current observation. In Givs, we choose both $\alpha = \beta = 0.5$ to value observations equally.

b: ESTIMATION ERROR METRIC

Suppose there is a given time series data $\{x_1, x_2, \cdots\}$ and we use a smoothing technique to generate a k-step-ahead estimate of this series $\{\widehat{x}_1, \widehat{x}_2, \cdots\}$. Note that the first k estimates are blind estimates since the forecaster does not have any observations yet. The (t + k)-th estimate is generated based on the t-th observed value (i.e., x_t). We use the Mean Absolute Error (MAE) to characterize the estimation error, which is defined as:

$$MAE = \mathbf{E}\left[|x_i - \widehat{x}_i|\right], \tag{4}$$

MAE captures the absolute distance between estimated data points and observed ones. In Givs, it indicates the average distance between the forecasted cursor position and actual position.

3) FINGER TAPPING DETECTION

Clicking is an essential operation for controlling mobile devices. Givs interprets index finger tapping as a clicking command, as shown in Figure 5. Leap Motion provides a built-in algorithm to detect tap gestures in office environments. The built-in algorithm tracks the absolute velocity of the index finger tip and recognizes the velocity change pattern during tapping. Figure 6 shows the Y velocity for the index finger tip (v_{index}) while performing a perfect tap gesture. The tap gesture starts with from zero v_{index} when the index finger aligns with the rest fingers. Then the index finger tip accelerates downwards and then decelerates until it reaches the lowest point. Finally, the finger tip restores to the original position with an opposite velocity change. To accurately identify finger tapping based on velocity in driving environments, we need to address the following challenges:

1) Several driving factors such as car vibration, hand shaking and bumps could affect the velocity of finger tips, which add notable noises to affect pattern recognition accuracy.
2) Latency of tapping detection algorithm is curcial for user experience. We require to have a detection delay that will not be perceptible by users.
3) To reduce latency, the detection algorithm must detect the gesture before it is completely finished. This requires careful parameter tuning to reduce false positives.

Algorithm 1 illustrates our tap detection algorithm, which updates its gesture recognition result once it receives a frame. It takes the v_{index} and the velocity of the middle finger tip v_{middle} as inputs and reports true if a tap gesture is detected. To address the above-mentioned challenges, we propose the following techniques.

a: DIFFERENTIAL METHOD FOR MITIGATING NOISES

Our recognition algorithms relies on velocity tracking to recognize gestures. This requires accurate measurement of the velocity. However, in a driving environment, the index finger tip velocity is affected by many factors, including the car vibration (Δ_v), biological shaking of fingers (Δ_d) and occasional cases of encountering bumps. Thus the raw tip velocity is fairly noisy and sensitive to road conditions.

Algorithm 1 The Index Finger Tapping Recognition Algorithm

```
Input:
  $v_{index}$: the y-direction velocity of the index finger
  $v_{middle}$: the y-direction velocity of the middle finger
  $W$: the length of the sliding window (constant)
function INIT
  $skipcounter \leftarrow 0$
  $q \leftarrow newQueue(W)$
end function
function ONFRAME($v_{index}, v_{middle}$)

  $rv \leftarrow v_{index} - v_{middle}$
  $q.add(rv)$
  if $skipcounter > 0$ then
    $skipcounter \leftarrow skipcounter$-1
    return false
  end if
  if not $|v_{middle}| < V$ then
    return false
  end if
  $\hat{rv} \leftarrow min(q)$
  if $V_{lim} < \hat{rv} < V$ and $\hat{rv} > V$ then
    $skipcounter \leftarrow W$
    return True
  else
    return True
  end if
end function
```

We observe that these environmental noises affect not only the index finger but also all other finger tips. Thus, we propose to use the differential method to effectively null out these noises. More specifically, instead of using the absolute velocity of the index finger, we use the relative velocity of the index finger tip with respect to middle finger tip as our decision variable, denoted as $v_{index|middle}$ or r_v. By doing so, common mode that appears on both index and middle finger (correlated noises) can be eliminated. **Tracking window and thresholds tuning.** When the index finger is at the lowest point, we define the distance between the index and middle finger tips as the tapping distance, denoted as d.We also define ttap as the time to complete the tap gesture, i.e., tapping time. Before discussing our various thresholds, we first quantitatively define a tap gesture as follows:

1) The tapping distance must be larger than a constant D. In practice, we choose $D = 3cm$.
2) The index finger then restores to almost the original position.
3) The entire gesture is completed within two time bounds, i.e. $T_{min} < t_{tap} < T_{max}$. In practice, we choose $T_{min} = 100$ms; $T_{max} = 300$ms based extensive user habit study.

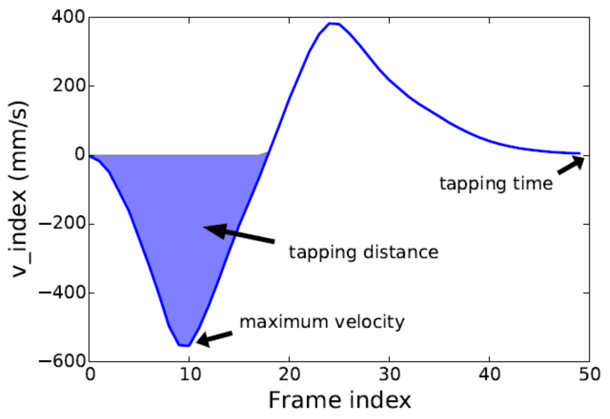


FIGURE 6. The velocity change pattern for a tap gesture.

In Figure 6, we observe that the finger tip has a notable maximum velocity along -Y axis, denoted as (v_{max}). Our algorithm detects this maximum velocity along with a decreasing trend to identify the tapping pattern.

With a known tapping distance d and tapping time t_{tap}, we have the following relations:

$$\begin{aligned} -d &= \int_0^{\frac{1}{2}T_{tap}} rvdt \\ &= -\int_0^{\frac{1}{2}T_{tap}} v_{max}\sin(\frac{2\pi}{T_{tap}}t)dt \\ &= \frac{|v_{max}\cdot T_{tap}|}{\pi}. \end{aligned} \quad (5)$$

Thus the velocity has a bound given by

$$|v_{max}| = \frac{\pi d}{T_{tap}} \geq \frac{\pi D}{T_{max}}, \quad (6)$$

With the constant settings above, we derive the threshold for v_{max} at approximately 190mm/s, denoted as V.

With this given threshold, the tapping detection algorithm keeps tracking r_v in last W frames (in a sliding window manner) to find one that exceeds the threshold. Once such a data point is found, the algorithm waits until the velocity returns above that threshold (velocity decreasing trend). Then this velocity change pattern is identified as finger tapping. The sliding window length W should be large enough to cover half of the pattern length, i.e.,

$$W \geq max\frac{1}{2}T_{tap} = \frac{1}{2}T_{tap} = 0.25s \approx 15frames, \quad (7)$$

b: RESPONSIVENESS TO RECURRING TAPPING

For each tap, the detection algorithm should report exactly once. Thus, after our algorithm detects one tap gesture, it waits for a full length of W frames before reporting new taps. At the end of W frames, the index finger should restore to the original position. Note that if a tap is detected, it's reported to Givs instantly without waiting. As a result, a tap gesture may be detected before the user finishes tapping, and latency is reduced significantly. Touchscreen based systems cannot achieve the same since tap and drag share the same staring point - finger down. While in Givs, tap and drag are interpreted from completely different gestures, and instant decision becomes possible.

c: MINOR ANOMALY ELIMINATION

In addition to the abovementioned mechanisms, two more approaches are used to filter out anomalies that could happen in real driving environments. First, we require that the middle finger does not move much when the index finger is tapping, and we add a filtering condition to bound v_{middle} between $\pm V$. Second, we observe that sometimes the velocity might encounter a glitch (a sudden change that is not possible with human speed). To address this problem, we further require r_v to be smaller than V_{lim}, which is the fastest possible movement speed of finger tips. This value is measured with real experiments at approximately 1000mm/s.

C. DELAY-SENSITIVE CONTROL ALGORITHM

The gesture recognition module of Givs works in an event-driven manner. When the recognition module receives a frame from the motion sensor, it updates the recognition results accordingly. The control algorithm is responsible for translating gestures to touchscreen commands and issuing these commands in the same sampling rate of a real touchscreen. In most cases, the gesture recognition output rate is slower than the touchscreen sampling rate. This could result in lags (e.g. the trail dragging an icon becomes laggy instead of being smooth) and sometime strange activities (e.g. the navigation app suddens jump to the other side of the earth). Givs addresses these critical and practical issues with a careful delay deadline.

1) A VIRTUAL TOUCH SCREEN

Android devices leverage standard Linux *evdev* input system [17] to process input events. The mutli-touch screen hardware reports its states by injecting evdev commands into the system's input event stream. A real touchscreen event stream comprises 2D coordinates of the touched point, touching pressure and timestamps for each dots. Givs maps palm center position (three coordinates in Leap Motion's metric coordinate system) to a 2D plane (touch screen 2D space in pixels, eliminating Leap Motion's Y axis). For pressing pressure, Givs always keeps a fixed value.

2) DELAY MANAGEMENT

Proper timing is the most important factor for Givs to work seamlessly exactly like a real touchscreen. The actual multi-touchscreen component emits input events in a fixed rate (with fixed delay). We define this as the control period T_{ctl} for an Android device. We obtain this value by capturing the evdev trace when the user is performing touchscreen operations. According to our measurements, $T_{ctl} \approx 20ms$. Thus Givs control algorithm must generate one control signal during each $20ms$ interval. However, as we have seen in

Section 2.5, only 64% of the frames arrive within the 20*ms* deadline. To overcome this, our control algorithm performs interpolation (based on forecasting) to overcome delay jitters. More specifically, for the activating and adjusting gesture, we simply use the last observed data points to forecast the next one since both gestures reflect a particular hand state that will not change in a short time. The Holt's method introduced above is used to forecast the missing values for moving gestures. Interpolation is not required by tap gesture since the tap recognition itself is resilient to delay jitters and will properly report a tapping once it happens.

D. BUMP DETECTION

When the car drives through bumpy or rough roads, users should focus on roads instead of using gestures. At the mean time, these special road conditions could easily distort hand gesture, which suggest gestures are not proper at that time. Givs advocates users to keep hands on the wheel for such circumstances and only use wheel buttons for limited but safer interactions (e.g. turn on light, mute the audio system, etc). For this purpose, Givs leverages prevailing bump and pothole detection techniques to detect such conditions, disable gesture recognition and warn the user if he/she is still performing gestures. Givs detects bumps and potholes based on z-peak bump detectors [9], [25] that detect spikes in z acceleration to identify bumps. According to previous measurements [25], with a threshold of 1:45g, z-peak is sensitive to road turbulence. If the turbulence reaches a given threshold, Givs will not feed sensor inputs to recognizers but instead warn the user if hand is still present in the sensor's view.

IV. EVALUATION

Our evaluation includes the benchmarks for individual gesture recognition algorithms and case studies to control real Android applications in various driving environments.The video demos could be accessed from Youtube [19].

A. METHODOLOGY

Our evaluation involves 8 male and 2 female participants, with each experiment lasting for 40 minutes on average. The leap motion sensor provides a set of built-in gesture recognizers, with the capability to track hand position and detect finger taps, but mainly targeted office environments. We compare our enhanced algorithms with these two builtin cases to demonstrate how driving environments affect gesture recognition as well as our advantages under such environments.

1) DEVICE SETUP

For the motion sensor, we use the Leap Motion sensor along with its latest SDK and recognition algorithms version 2.1.5+22699. For the accelerometer, we use the on-board InvenSense MPU-6050 accelerometer of two Google Nexus 7 Tablets (2013 version). These accelerometers have a 50mg error along X and Y axis and a 80mg error along Z axis ("mg" is milli-g, with "g" denotes gravity). The device

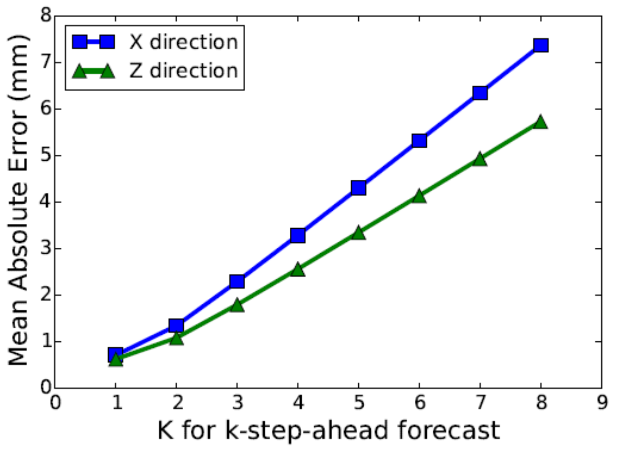


FIGURE 7. The MAE of k-step-ahead forecast for both X and Z positions.

being controlled is a Samsung Galaxy Nexus S with a screen resolution of 1280 x 720 pixels.

2) DRIVING SETUP

Our testing platform (vehicle) is an Audi Q5 SUV. Our road condition is an urban road with a length of 6km and a speed limit of 55km/h. Our driving scenarios include: 1) Fully stop with engine off, which simulates a vibration-free environment similar to the office. 2) Fully stop with engine on, which simulates a parking state. 3) Driving with even speed, which simulates a nor mal driving condition.

B. HAND POSITION TRACKING

Since Givs simulates a virtual cursor on device screen to track hands, the position tracking algorithm must accurately and constantly report the position updates. To meet this criteria, we use Holt's linear model to forecast hand positions when the motion sensor readings are not reliable (due to bumps) or delay jitters are encountered. In this part, we evaluate our position tracking algorithm from two perspectives. The first experiment aims to understand the estimation errors, which validates the accuracy of the forecast method. The second experiment illustrates how the forecast method helps to amortize delay jitters and the impact of bumps in real driving conditions.

1) FORECAST STEPS AND ERRORS

According to Table 1, the maximum frame delay could be as high as 80ms. If we would like to have a gesture control reaction latency less than 20ms, the Holt method needs to forecast at least 3 values to bridge this gap. Also when the car experiences bumps, Givs reverts to the Holt method to produce several estimates. We would like to quantitatively measure the errors for k- step-ahead forecasts. To obtain ground truth, we collect the raw hand position trace from the motion sensor in an office environment (with no vibration and bumps). Then we use the Holt method to perform k-step-ahead forecasts based on this time series data and compare the estimates with the raw readings (ground truth). Figure 7 presents the Mean Absolute Errors for k-step-ahead estimates. As expected,

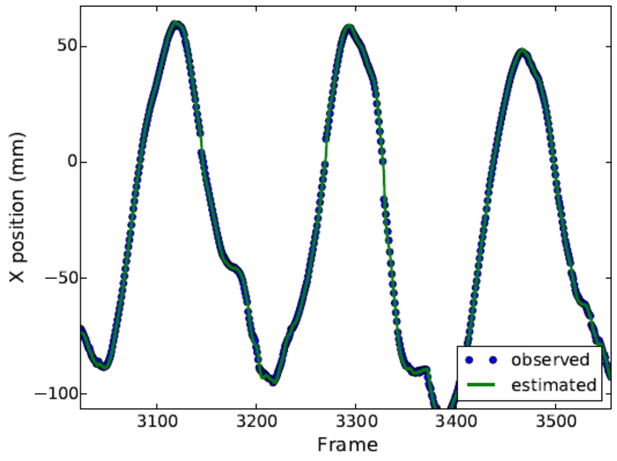


FIGURE 8. The one-step-ahead forecast of X-direction palm position. During three jitters, positions are estimated.

the error increases when we try to forecast more future frames. This error increase trend is nearly linear. In practice, the Holt method can generate up to 8 acceptable accurate estimates, with an MAE bounded within 8mm.

2) FORECASTING USEFULNESS

Figure 8 shows the X positions when a participant is moving hand along the X axis. We perform one-step-ahead forecast and the green line plots the estimates generated by our Holt forecaster. As shown in the graph, since hand movement speed is subject to human limits, the hand positions change smoothly with notable trends. Thus, our Holt forecaster can effectively capture this trend and accurately approximate the observed values. As shown in the graph, we observe two delay jitters at frame 3250 and 3350 respectively. By using the Holt estimates, we can effectively remove these two jitters. In addition, when bumps are detected (normally last for less than 5 frames), we can also use the Holt method to provide reliable forecasts.

C. TAP GESTURE DETECTION

We evaluate our tapping detection algorithm from two perspectives. On one hand, we characterize its detection accuracy (false positives and negatives) with different driving conditions. On the other hand, we characterize its responsiveness of the algorithm by measuring the time elapsed since the user performs the gesture until it is detected. We compare our algorithm with the Leap Motion built-in tapping detection algorithm.

1) TAPPING ACCURACY

Table 2 summarizes the performance of both algorithms across all participants. As shown, our algorithm outperforms the built-in algorithm with an average true positive rate of 90%. *male*1 performs tap gestures with an almost fist hand state, which makes it hard for the sensor to see the fingers and thus affect the performance of both recognition algorithms.

TABLE 2. Tap detection performance.

	Givs			Built-in		
Participant	%FP	%FN	Delay	%FP	%FN	Delay
male1	0%	24%	124.3	0%	92%	199.5
male2	0%	0%	132.9	0%	88%	161.7
male3	0%	2%	178.5	2%	66%	215.1
male4	0%	2%	156.3	4%	38%	209
male5	0%	2%	145.4	0%	34%	186.5
male6	0%	2%	136.3	12%	24%	169
male7	0%	0%	132.1	2%	12%	182.5
male8	0%	0%	136.2	0%	8%	194.7
female1	0%	4%	132.7	0%	48%	204.7
female2	0%	0%	109	2%	10%	160.5
Average	0%	3.60%	138.4	2.20%	42%	191.1

Figure 9a presents a scenario when a user is tapping while the vehicle drives through a bumpy road. The first graph shows the absolute Y-direction velocity of the index finger tip. We observe that the first tapping (from frame 0 to 100) is performed when the car meets a bump, which is largely affected by the bump. The following taps also observe lots of noises (due to Δ_b and Δ_v). The second graph shows the relative velocity of the index finger tip. As we have mentioned, correlated noises such as Δ_v and bump influence are effectively eliminated by this differential method. As a result, we observe much clear tapping patterns from the relative velocity. Finally, the third graph shows the detection results of both algorithms. Givs uses relative velocity as its decision variable and successfully identifies all the 8 taps. On the contrary, the built-in algorithm only detects four taps and four false positives (note there are two between frame 700 to 800). These false positives all happen immediately after the detecting one tap. Givs overcomes this problem by disabling itself for a whole tracking window after detecting a tap.

2) EFFECTIVENESS OF THE DIFFERENTIAL METHOD

The differential method can not only eliminate correlated noises but also avoid confusing other gestures with tapping. For example, the adjusting gesture requires the flat hand to lean forwards. Thus the absolute velocity of index finger when performing this gesture is very similar to tapping, as shown in Figure 9b. By using the relative velocity, Givs notices that both index and fingers are moving together and there is no relative velocity. Consequently, with the differential method, Givs will not detect any taps for this scenario.

Tapping detection delay. Table 2 also presents the delay for detecting a tapping action. In general, Givs relies on the relative velocity of the index finger tips which has less noises and more obvious trends when the user performs a tap. Thus, our algorithm effectively reports tapping earlier than the built-in algorithm that is based on the absolute index finger tip velocity. As shown in Table 2, our algorithm recognizes a tap gesture by 27.5% faster than the built-in algorithm on average. According to previous user experience study [23], the delay between performing a tap gesture and the visual feedback should be no more than 0.1 to 0.2 seconds. Also our detection delays are bounded within 160ms across

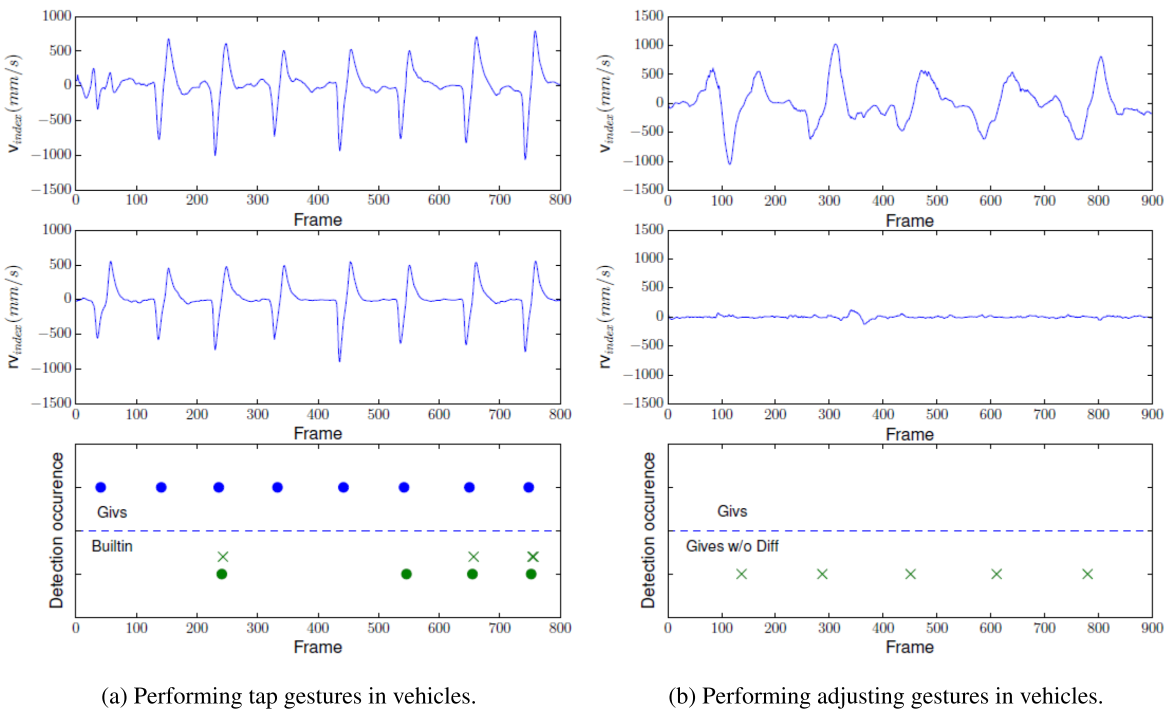


FIGURE 9. Tap detection for two scenarios while driving through bumpy roads. v_{index} stands for the absolute Y-direction velocity of the index finger. r_{vindex} denotes the index finger velocity relative to middle finger. In "detection occurrence", a dot represents a true positive while a cross stands for a false positive.

all participants except one while the delays of the built-in algorithm are all beyond 160ms. The out-liner user performs gestures much slower than others and thus both algorithms observe a higher detection delay. We will show that during the user study, our participants can sense the delay difference between our algorithm and the built-in in general.

D. DISTRACTION EXPERIMENTS

We invite our participants to several in-vehicle user studies by using gestures to control common driving-related applications. We design three applications: 1) a dialpad application where the cursor track user's hand and taps are used for inputing number; 2) a music playback interface similar to [15]; 3) a drag-and-drop interface where the user can drag random icons to delete them. The experiment setup is shown in Figure 1.

1) DIALPAD APPLICATION

For this application, users report false positives and negatives for Givs as compared to Leap Motion's built-in tap recognizer. The result is in accordance with our evaluation results shown in Table 2, where the built-in algorithm produces 2x to 10x more false negatives across our users. This shows that existing gesture recognizers intended for indoor usage do not perform good because of various noises in a driving environment.

2) MUSIC PLAYBACK

For the music playback application, we measure the time that a participant turns away from the front road, denoted as

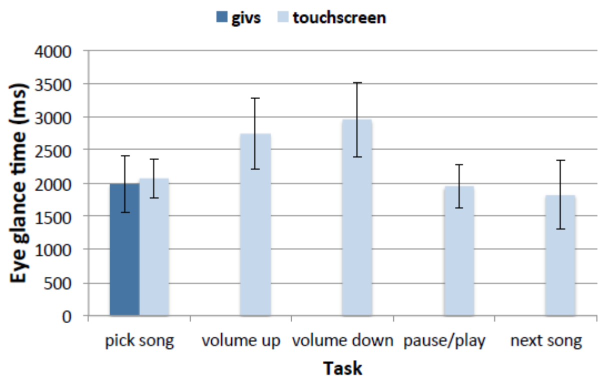
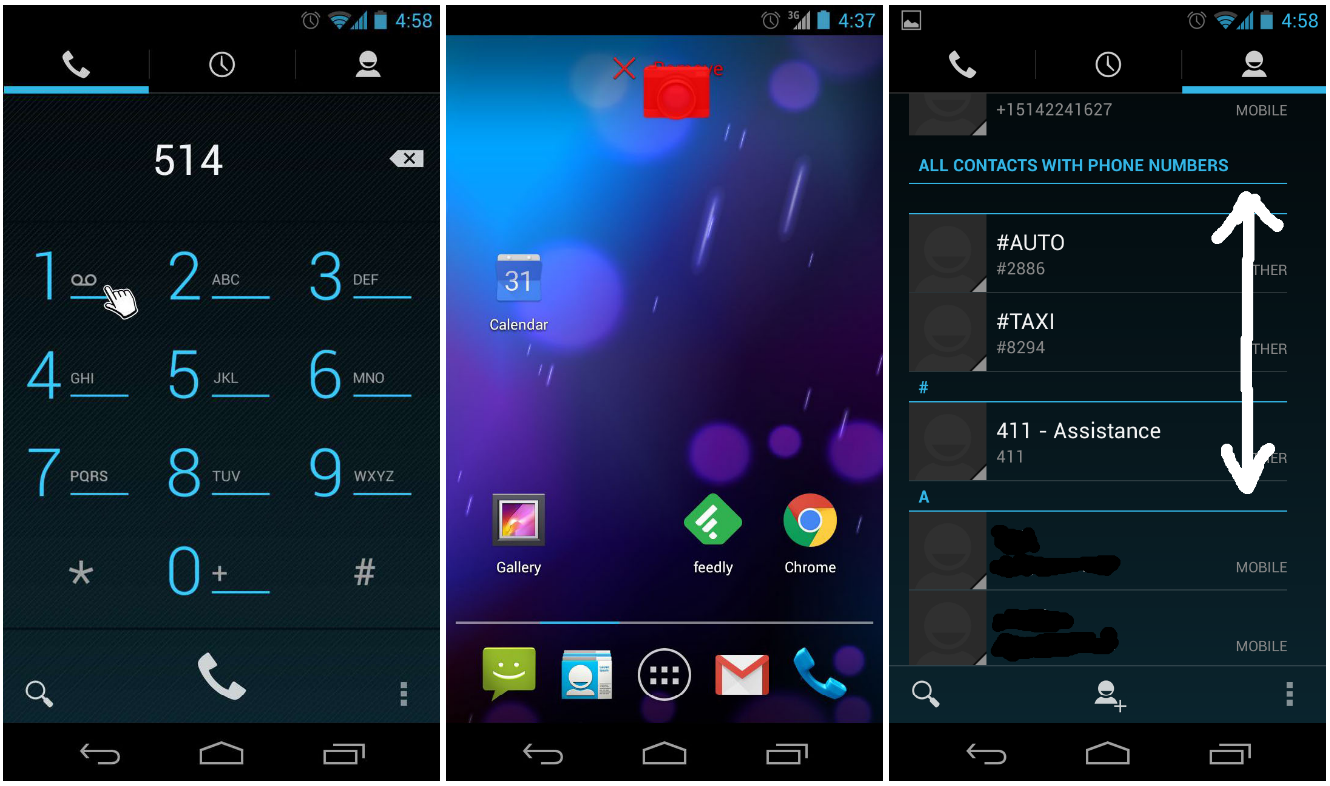


FIGURE 10. Eye glance time for gesture and touchscreen interactions.

the eye glance time. This index quantifies the distraction caused by the interaction method. We test five tasks commonly performed for a music playback application while driving [15]. We compare Givs with traditional touch screen interactions. The mobile device is placed at the car dashboard as in in Figure 1.

Figure 10 shows the results. We have the following findings:

1) Traditional touch screen requires full visual interactions for any task while gesture control does not impose this requirement (only one for this particular application). As a result, gesture control allows the vehicle users to keep eye on the front road and reduces the user distraction.



(a) Input phone number via a dialpad. (b) Remove random icons on screen. (c) Scroll to browse contact list.

FIGURE 11. In-vehicle user experience study scenarios.

2) For tasks that must have visual interaction, gesture control shortens the movement distance of the hand. This helps vehicle users to hold the steering wheel stably, and thus reduces user distraction.
3) While driving, tapping a small area on a touchscreen requires more attention than performing a gesture anywhere in the 3D space, incurring more cognitive distraction.

3) DRAG-AND-DROP

This test stresses the hand tracking and state tracking accuracy of Givs. We require participants to score the user experience (score 10) and provide suggestive feedbacks. The results show that over 60% of the participants are satisfied with the drag-and-drop experience and report a score over 8. The remaining 30% report that the above drop area is too small to be used in a driving environment. This observation opens an suggestion for apps to use larger UI elements and bigger fonts in a driving environment. For example, the Android system provides system-wise adjustment of font and UI element sizes, which could be very useful in driving environments.

E. USER EXPERIENCE EXPERIMENTS

We invite our experiment participants to perform three common mobile usage scenarios for our user experience experiment. We intend to use these case studies to demonstrate the user experience of using Givs gestures, by using different scenarios to stress test different parts of our system.

1) CASE STUDY 1

Dialpad and tapping. The first use case involves a typical phone call dialpad, as shown in Figure 11. Givs displays a virtual cursor that tracks participant's hand. Participants will perform tap gestures to input his/her 10- digit phone number for four times (two with Givs and two with the built-in algorithm). We randomly generate the sequence for Givs and the built-in to avoid the bias that participants know which algorithm he/she is testing with. This test stresses the tapping and hand tracking capability of Givs.

During this test, our participants report subjective feedbacks along with three objective metrics for tapping recognition: 1) number of retries (false negatives): the algorithm does not recognize a tap performed by the user; 2) unintended inputs (false positives): the system inputs a digit but the user does not tap or the system inputs a digit multiple times when the user only taps once 3) wrong taps: the system inputs a wrong digit when the user taps. We also require participants to note down whether he/she can feel a lag when the system responds to the tap gesture.

The false positive/negative statistics for Givs and the built-in algorithm is in accordance with our evaluation results shown in Table 2. The built-in algorithm produces 2x to 10x more false negatives across our users. This is because the noises and bumps in driving environment largely affects the index finger velocity and damages the performance of the built-in algorithm dependent on it. One user fails to finish this test with the built-in algorithm due to too many false negatives. From participants' feedbacks, we learn that users treat false positive and negative cases as the problem with the system while treating wrong taps sometimes as their own faults. We conclude from our survey that by minimizing false positives and negatives, Givs can greatly improve the user experience for inputing.

As for the delay awareness, around 50% of our participants report noticing the lags when using the built-in algorithm. No users report lag problems with Givs.

2) CASE STUDY 2

Drag and drop. The second use case involves a home screen with randomly placed icons, as shown in Figure 11b. Participants use the moving gesture to drag and drop all the icons to the top of the screen so as to remove them. This test stresses the hand tracking and state tracking functionality of Givs. We require participants to score the user experience (score 10) and provide suggestive feedbacks. The results show that over 60% of the participants are satisfied with the drag-and-drop experience and report a score over 8. The remaining 30% report that the above drop area is too small to be used in a driving environment. This observation opens an suggestion for apps to use larger UI elements and bigger fonts in a driving environment. For example, the Android system provides system-wise adjustment of font and UI element sizes, which could be very useful in driving environments. This case study shows that simplifying UI design would also help user experience when new interaction methods such as gesture control are introduced.

3) CASE STUDY 3

Scrolling. The third use case presents user's contact list. Participants use the adjusting gesture to scroll and browse the list, as shown in Figure 11c. This test mainly address the adjusting state tracking of Givs. We require all participants to perform several gestures to get familiar with the leaning angle of the system. Then all participants will perform scrolling in the vehicle environments and score their user experience (score 10). We observe that all users except one are satisfied with the default parameters. Only one user reports that the angle limit is too small and sometimes scrolling is activated when not intended. We double the angle limit and redo the same procedure for this user again. With the new angle limit, the previous concern of the user is eliminated. We conclude from this test that the angle limit is a subjective parameter to different users. In real deployment, we will provide the default limit while allow allow users to change this limit based on their preferences.

V. RELATED WORK

Both the industry and research community have been actively developing safe and convenient interaction methods for complicated real driving environments. Existing approaches include tactile interactions, touchscreen, voice, touch-based gestures and touch-less gestures. The tactile interaction was the most popular technique where the user interacts with the car system based on a set of physical buttons embedded in the wheel or dashboard. According to the study [15], this technique requires considerable eye focuses and is largely replaced by newer technologies.

A. VOICE CONTROL

Voice control is also a popular and widely adopted technology. Users issue voice commands, which are recognized by the system to perform a fixed set of operations. Voice control has been proven useful to input texts or perform defined operations such as answering a phone call [13], [14]. For instance, systems like Siri [4], Google Assistant [10] and Mancuso [21] largely relies on voice recognition to input texts and search destinations in map navigation. Though voice control is useful in the abovementioned cases, it still fails to cover all the important use cases in a vehicle. For example, users cannot use voice to smoothly move the map area of a navigation application.

B. TOUCH-BASED GESTURE CONTROL

Gestures provide a natural and safe way for human-computer interaction. Existing gesture control approaches can be classified based on its underlying technology. Touch-based approaches normally embed a touch screen for user to perform gestures with touching [8], [29]. These approaches are largely constrained by the size and location of touchscreen. The steering wheel can only equip a small screen, but anywhere else pose a significant inconvenience for touching.

C. TOUCH-FREE GESTURE CONTROL

To avoid being tied with a particular physical device, touch-free gestures are becoming increasingly popular. The most well developed approachs are based on computer vision [6], [27] that capture hand images and recognizes hand gestures. However, previous vision-based techniques are easily constrained by lighting conditions and fail to be used in vehicle. Though depth cameras and short-range radars can be deployed in the vehicle to against variable lighting conditions [26], [28], their computational costs are considerable and on-road noise issues in real driving environments are still not well addressed.

In this paper, Givs elects to leverage Leap Motion for the same purpose of providing fine-grained tracking in cars. Leap Motion serves as the last iteration of innovation in this area which leverages innovative sensing techniques to provide 3D tracking of hands. Leap Motion sensor intends to capture fine-grained hand gestures such as finger movements. Thus, it is widely adopted for fine-grained control such as digital musical instruments control [31], handwriting recognition [36] as well as stroke rehabilitation [18].

D. WiFi-BASED GESTURE CONTROL

In addition to computer vision based approaches, WiFi distortion [16], [37], [38] based approaches are gaining popularity in research community for gesture recognition and driving safety. However the state-of-the-art solutions using this technique can only provide coarse-grained tracking (treat a hand as a whole object) or have to redesign the baseband silicon for real-time signal processing.

E. WEARABLE HARDWARE BASED GESTURE CONTROL

We are also aware of techniques that use hand wearable devices for gesture recognition. However, existing approaches [11], [39] suffer from low accuracy and coarse granularity, which is not mature enough for vehicle environments.

VI. CONCLUSION

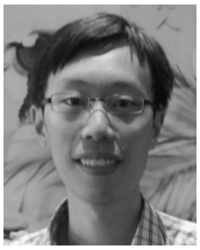
Hand-held devices have witnessed great success in the past few years but can hardly be used while driving due to safety and convenience limitations. In this paper, we systematically study these limitations and quantify the factors. To enable this important, we design and implement Givs. Various building blocks of Givs are redesigned and optimized for the noises, delays and special road conditions in driving environment. Our evaluation results show gestures can bring safety and convenience for various applications used in a car.We conclude that gestures serve as a promising replacement for existing in-vehicle interaction methods.

REFERENCES

[1] F. Adib and D. Katabi, "See through walls with WiFi!" in *Proc. ACM Conf. SIGCOMM*, Aug. 2013, pp. 75–86.
[2] Android. *Android Auto*. Accessed: 2019. [Online]. Available: https://www.android.com/auto/
[3] Apple. *Apple Carplay*. Accessed: 2019. [Online]. Available: https://www.apple.com/ca/ios/carplay/
[4] Apple. *Siri*. Accessed: 2019. [Online]. Available: https://www.apple.com/ca/siri/
[5] Governors Highway Safety Association. *Distracted Driving Laws, U.S.* Accessed: 2019. [Online]. Available: https://www.ghsa.org/index.php/resources/news-releases/ga-handheld18
[6] M. Bhuiyan and R. Picking, "Gesture-controlled user interfaces, what have we done and what's next," in *Proc. 5th Collaborative Res. Symp. Secur., e-Learning, Internet Netw. (SEIN)*, Darmstadt, Germany, 2009, pp. 25–29.
[7] *Diver Communications Gestures*. Accessed: 2019. [Online]. Available: https://en.wikipedia.org/wiki/Diver_communications
[8] T. Döring, D. Kern, P. Marshall, M. Pfeiffer, J. Schöning, V. Gruhn, and A. Schmidt, "Gestural interaction on the steering wheel: Reducing the visual demand," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2011, pp. 483–492.
[9] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: Using a mobile sensor network for road surface monitoring," in *Proc. 6th Int. Conf. Mobile Syst., Appl., services*, 2008, pp. 29–39.
[10] Google. *Google Assistant*. Accessed: 2019. [Online]. Available: https://assistant.google.com/
[11] J. Gummeson, B. Priyantha, and J. Liu, "An energy harvesting wearable ring platform for gestureinput on surfaces," in *Proc. 12th Annu. Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, 2014, pp. 162–175.
[12] J. Guna, G. Jakus, M. Pogačnik, S. Tomažič, and J. Sodnik, "An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking," *Sensors*, vol. 14, no. 2, pp. 3702–3720, 2014.
[13] Z. Hua and W. L. Ng, "Speech recognition interface design for in-vehicle system," in *Proc. 2nd Int. Conf. Automot. User Inter. Interact. Veh. Appl. (AutomotiveUI)*, 2010, pp. 29–33.
[14] R. W. Huang, P. Vaghefinazari, and S. Yamamoto, "Communication system and method between an on-vehicle voice recognition system and an off-vehicle voice recognition system," U.S. Patent 9 263 058, Feb. 16 2016.
[15] M. G. Jæger, M. B. Skov, and N. G. Thomassen, "You can touch, but you can't look: Interacting with in-vehicle systems," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2008, pp. 1139–1148.
[16] B. Kellogg, V. Talla, and S. Gollakota, "Bringing gesture recognition to all devices," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 303–316.
[17] *Linux Kernel Multi-Touch (MT) Protocol*. Accessed: 2019. [Online]. Available: https://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt
[18] M. Khademi, H. M. Hondori, A. McKenzie, L. Dodakian, C. V. Lopes, and S. C. Cramer, "Free-hand interaction with leap motion controller for stroke rehabilitation," in *Proc. Extended Abstr. 32nd Annu. ACM Conf. Human Factors Comput. Syst. (CHI EA)*, 2014, pp. 1663–1668.
[19] McGill CPSI Lab. *Gesture Control in Vehicles*. Accessed: 2019. [Online]. Available: https://www.youtube.com/playlist?list=PLQpCgZtmoc1IELmJU4yWZhAL1VmYlvouG
[20] Y. Ma, G. Zhou, and S. Wang, "WiFi sensing with channel state information: A survey," *ACM Comput. Surv.*, vol. 52, no. 3, p. 46, 2019.
[21] V. Mancuso, "Take me home: Designing safer in-vehicle navigation devices," in *Proc. Extended Abstr. Hum. Factors Comput. Syst. (CHI)*, 2009, pp. 4591–4596.
[22] Microsoft. *Azure Kinect DK*. Accessed: 2019. [Online]. Available: https://azure.microsoft.com/en-us/services/kinect-dk/
[23] R. B. Miller, "Response time in man-computer conversational transactions," in *Proc. AFIPS Fall Joint Comput. Conf.*, 1968, pp. 267–277.
[24] *Mirrorlink*. Accessed: 2019. [Online]. Available: http://mirrorlink.com/
[25] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. 6th ACM Conf. Embedded Netw. Sensor Syst.*, 2008, pp. 323–336.
[26] P. Molchanov, S. Gupta, K. Kim, and K. Pulli, "Multi-sensor system for driver's hand-gesture recognition," in *Proc. 11th IEEE Int. Conf. Workshops Autom. Face Gesture Recognit. (FG)*, May 2015, pp. 1–8.
[27] E. Ohn-Bar and M. M. Trivedi, "Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2368–2377, Dec. 2014.
[28] F. Parada-Loira, E. Gonzalez-Agulla, and J. L. Alba-Castro, "Hand gestures to control infotainment equipment in cars," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 1–6.
[29] B. Pfleging, S. Schneegass, and A. Schmidt, "Multimodal interaction in the car: Combining speech and gestures on the steering wheel," in *Proc. 4th Int. Conf. Automot. User Inter. Interact. Veh. Appl. (AutomotiveUI)*, 2012, pp. 155–162.
[30] X. Shen, G. Hua, L. Williams, and Y. Wu, "Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields," *Image Vis. Comput.*, vol. 30, no. 3, pp. 227–235, Mar. 2012.
[31] E. S. Silva, J. A. O. D. Abreu, J. H. P. D. Almeida, V. Teichrieb, and G. L. Ramalho, "A preliminary evaluation of the leap motion sensor as controller of new digital musical instruments," in *Proc. SBCM Brazilian Symp. Comput. Music*, Brasilia, Brazil, Oct./Nov. 2013, pp. 59–71.
[32] W. C. Stokoe, Jr., "Sign language structure: An outline of the visual communication systems of the american deaf," *J. Deaf Stud. Deaf Edu.*, vol. 10, no. 1, pp. 3–37, 2005.
[33] J. S. Supančič, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan, "Depth-based hand pose estimation: Methods, data, and challenges," *Int. J. Comput. Vis.*, vol. 126, no. 11, pp. 1180–1198, Nov. 2018.
[34] P. M. Swamidass, Ed., *HOLT' Forecasting Model*. Boston, MA, USA: Springer, 2000, p. 274.
[35] R. H. Venkatnarayan, G. Page, and M. Shahzad, "Multi-user gesture recognition using WiFi," in *Proc. 16th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2018, pp. 401–413.
[36] S. Vikram, L. Li, and S. Russell, "Handwriting and gestures in the air, recognizing on the fly," *Proc. CHI*, vol. 13, pp. 1179–1184, Apr. 2013.
[37] F. Wang, J. Liu, and W. Gong, "Wicar: WiFi-based in-car activity recognition with multi-adversarial domain adaptation," in *Proc. Int. Symp. Qual. Service*, 2019, p. 19.

[38] G. Wang, Y. Zou, Z. Zhou, K. Wu, and L. M. Ni, ''We can hear you with Wi-Fi!'' *IEEE Trans. Mobile Comput.*, vol. 15, no. 11, pp. 2907–2920, Nov. 2016.
[39] J. Wang, D. Vasisht, and D. Katabi, ''RF-IDraw: Virtual touch screen in the air using RF signals,'' *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 235–246, 2015.
[40] W. Zeng, C. Wang, and Q. Wang, ''Hand gesture recognition using leap motion via deterministic learning,'' *Multimedia Tools Appl.*, vol. 77, no. 21, pp. 28185–28206, Nov. 2018.

LANDU JIANG received the B.Eng. degree in information security engineering from Shanghai Jiao Tong University, the master's degree in construction management, the M.Sc. degree in computer science from the University of Nebraska–Lincoln, and the Ph.D. degree from the School of Computer Science, McGill University. He is currently a Mitacs Accelerate Industrial postdoctoral position with Aerial Technology and McGill University. His research interests include computer vision, machine learning, smart sensing, wearable and mobile computing, cyber-physical systems, green energy solutions, and online social networks.



MINGYUAN XIA received the bachelor's degree from Shanghai Jiao Tong University and the Ph.D. degree from McGill University. His research interests include applying program analysis for automated testing, bug finding, security checking, and optimization for mobile applications and systems. He is also interested in optimizing the throughput for distributed storage and distributed computing services.

XUE LIU (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign.

He has also worked as the Samuel R. Thompson Chaired Associate Professor with the University of Nebraska–Lincoln and HP Labs, Palo Alto, CA, USA. He served as the Chief Scientist of Tinder, Inc. He is currently a Full Professor with the School of Computer Science and a William Dawson Scholar (Chair Professor) with McGill University. He is also the Vice President R&D, a Chief Scientist, and the Co-Director of the Samsung AI Center Montreal. He is also a Professor (courtesy appointment) of mathematics and statistics with McGill University. He is also a Faculty and an Associate Member of the Montreal Institute for Learning Algorithms (MILA). He has published more than 280 research articles in major highly-reputable peer-reviewed international academic journals and conference proceedings, including ACM MobiCom, IEEE INFOCOM, ICNP, IEEE Security and Privacy (Oakland), IEEE RTSS, IEEE RTAS, ACM/IEEE ICCPS, WWW, ACM UbiComp, ACM KDD, and IEEE ICDE, and various IEEE/ACM Transactions. His research interests are in computer and communication net-works, real-time and embedded systems, distributed systems, cyber-physical systems, green computing, and smart energies. He served on various national and international grant review committees or panels. He is or had been an Editor/Associate Editor of the IEEE/ACM Transactions on Networking (ToN), *ACM Transactions on Cyber-Physical Systems* (TCPS), the IEEE Transactions on Vehicular Technology (TVT), the IEEE Communications Surveys and Tutorial (COMST), and the IEEE Transactions on Parallel and Distributed Systems (TPDS).

FAN BAI received the B.S. degree in automation engineering from Tsinghua University, Beijing, China, in 1999, and the M.S.E.E. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, in 2005.

He has been a Staff Researcher with the Electrical&Control Systems Laboratory, Research and Development and Planning, General Motors Corporation, since September, 2005. He published 90 research articles in top-quality conferences and journals, particularly, INFOCOM, SECON, Mobicom, Mobihoc, Sensys, IEEE JSAC, IEEE TMC, IEEE/ACM TON, IEEE TVT, and IEEE TWC. In addition, he published one book and six book chapters. His publications received more than 8 000 citations (according to Google Scholar). He also has more than 80 patents granted or pending. His current research is focused on the discovery of fundamental principles and the analysis and design of protocols/systems for next-generation vehicular networks, for safety, telematics, and infotainment applications.

Dr. Bai received the Charles L. McCuen Special Achievement Award from General Motors Corporation in recognition of his accomplishment in area of vehicle-to-vehicle communications for drive assistance and safety. He was featured as ITS People in 2014 by the *IEEE ITS Magazine* for his technical contributions to vehicular networks and intelligent transportation systems. He serves as the Technical Program Co-Chair for IEEE WiVec 2007, IEEE MoVeNet 2008, ACM VANET 2011, and ACM VANET 2012, among other leading roles in academic and industry technical conferences. He is an Associate Editor of the IEEE Transaction on Vehicular Technology and the IEEE Transaction on Mobile Computing. He also serves as a Guest Editor for the *IEEE Wireless Communication Magazine*, the *IEEE Vehicular Technology Magazine*, and *AdHoc Networks* Journal (Elsevier). He is also serving as a Ph.D. Supervisory Committee Member at Carnegie Mellon University, University of Illinois–Urban Champaign, and University of Southern California. He is Distinguished Lecturer of IEEE.