# Rulers on Our Arms: Waving to Measure Object Size through Contactless Sensing

YANG LIU, YONGHANG JIANG, ZHENJIANG LI, and JIANPING WANG,
City University of Hong Kong, China

In this article, we propose a mobile system, *Aware*, which turns our wearable or mobile device into a ruler. It can estimate the size of objects that could be large in size and not directly touchable by the user. Such a design will enable a rich set of applications that count on the size information of surrounding environments/objects. *Aware* purely utilizes the motion sensors on the device for object size measures. It can also integrate with the crowdsourcing feature for both performance improvement and result sharing. We propose a series of key techniques to address three major challenges in the *Aware* design: (1) user's angle of line-of-sights to the object is used in the size measure but motion sensors track only the angle of arm's waving, (2) motion sensors are noisy that require novel and effective data processing techniques, otherwise the errors could easily overwhelm the final result, and (3) in the crowdsourcing mode, *Aware* needs to identify vicinal objects of similar sizes and effectively fuse the measured sizes that correspond to the same object. We consolidate the above designs and implement *Aware* on Android platforms. Extensive experiments with four users show that *Aware* can achieve accurate measurement performance for the objects of various sizes in both indoor and outdoor environments.

CCS Concepts: • **Human-centered computing → Mobile devices**;

Additional Key Words and Phrases: Object size measure, mobiles, wearables, contactless sensing

**ACM Reference format:**
Yang Liu, Yonghang Jiang, Zhenjiang Li, and Jianping Wang. 2019. Rulers on Our Arms: Waving to Measure Object Size through Contactless Sensing. *ACM Trans. Sen. Netw.* 15, 1, Article 14 (February 2019), 25 pages.
https://doi.org/10.1145/3289183

## 1 INTRODUCTION

This article presents *Aware*, a system that turns our wearable or mobile device, e.g., smart watch or phone [31], into a virtual ruler. The design of *Aware* is inspired by the scenario envisioned in the future—the size information of objects on the earth, e.g., height and width, becomes widely available, which is attached to the object in the digital space and can be easily displayed or extracted by the user, e.g., through street view softwares, like Google Maps. We foresee this can benefit at least:



This work is partially supported by the ECS grant from Research Grants Council of Hong Kong (Project No. CityU 21203516), the GRF grant from Research Grants Council of Hong Kong (Project No. CityU 11217817), and the grant from Science Technology and Innovation Committee of Shenzhen Municipality (Project No. JCYJ20170818095109386).
Authors' addresses: Y. Liu, Y. Jiang, Z. Li (corresponding author), and J. Wang, Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon Tong, Hong Kong, China; emails: {yliu562-c, yhjiang4-c}@my.cityu.edu.hk, {zhenjiang.li, jianwang}@cityu.edu.hk.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
1550-4859/2019/02-ART14 $15.00
https://doi.org/10.1145/3289183

(1) *Working convenience related to the object size.* Decoration workers can browse their customer's house and yard sizes in advance on-line to decide the appropriate volume of materials (e.g., oil paint) and suitable instruments (e.g., ladder) to bring. Truck drivers can check the height limits before the departure to select the best route. Designers can decide the proper poster size sticked on different places for charity or social activities.
(1) *Smart living.* Based on the height of an apartment in the building, the indoor temperature and lighting (affected by the sunshine) and the air circulation (affected by the air pollution level outdoor) can be controlled intelligently. The object sizes could also be seamlessly integrated into the Augmented reality (AR) [10], e.g., for famous objects and architectures, to benefit tourist and education, e.g., in history and geography classes.
(1) *Study activities.* Students or researchers, e.g., in biology, could collect enriched experimental data. For instance, the biology students can record a regular growth process of large plants, e.g., height and width of the crown. In addition, some very detailed information, like the distribution of fruits on the plant, e.g., fruits' heights, can also be efficiently recorded from the crowds for study.

Although this might be an intriguing scene about the future, it has unveiled an emerging and desired need to measure object size, e.g., height and width, where the measured area can be large in size and not directly touchable by the user, e.g., far away or hanging in the air. In addition, the solution needs to be **convenient**, such that people can easily perform the measurements and be willing to contribute their results to the envisioned digital space, which is clearly difficult to be built by any centralized authority due to the large object volume in our daily life.

In the literature, there are prior efforts that can measure object size from a large amount of images [6, 17] in the research community, but the computation normally needs to be offloaded to the server, not afforded by the portable platforms along with the user, e.g., smart watch or smart phone, since these works utilize expensive computer vision algorithms in the design. As the computation is not conducted locally, the offloaded images will also increase the transmission overhead and incur potential privacy concerns, largely alleviating users' willingness to measure and share their results. Although some recent designs [3, 19] can execute on mobile devices locally but suffer the limitation that the measured area must touch on the ground and/or only the height measurement is viable. In addition, these methods also suffer the issues, including the impact of ambient light condition, the camera distortion, the difficulty of calibration, and so on. In summary, the limitations stated above fundamentally prohibit existing designs being a convenient and desired solution as envisioned above.

To cope with these drawbacks, in this article, we introduce *Aware*, a contactless object size sensing service design using purely motion sensors, e.g., accelerometers and gyroscope, to fulfill the aforementioned need. To measure the size from $a$ and $b$ of an object as in Figure 1, a user straightens his arm and waves from pointing at $a$ to $b$. Geometric relation (among various formed triangles) is then established between the user and the object, and their *relative* quantity can be derived using motion sensor readings. The user repeats this process at another location a few steps away, leveraging the moving distance as a bridge to convert the relative relation to the absolute size of the object. This principle is extensible for the object width measurement as well, and also compatible to the case when the measured area is hanging in the air, e.g., the size from $a$ to $c$ as in Figure 1. All *Aware*'s computations for the object size measure can be completed on the wearable or mobile device locally, while *Aware* can also work in an advanced mode with crowdsourcing for both the performance improvement and result sharing.

However, translating the above idea to a real system entails the following challenging issues:

(1) *Indirect measurements. Aware* adopts user's angle of line-of-sights to the object (e.g., from eyes to $a$ and $b$ in Figure 1) in the size measure. However, motion sensors track the angle

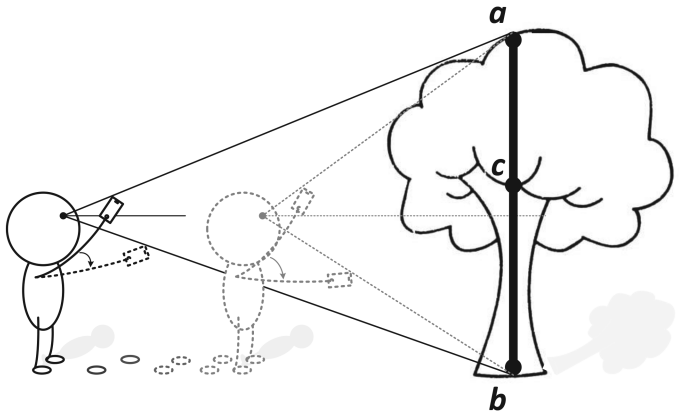


Fig. 1. ***Aware* 's working principle.** User waves the arm to measure the object size at two locations steps away, using wearable or mobile motion sensors.

of arm's waving, which is not equal to the angle of line-of-sights, since the user aims at the points using eyes. Nevertheless, how to extract this hidden relation, which can take such indirect angle inputs to derive the object size, is so far unknown yet.

(2) *Low-accuracy measurements.* In addition to the indirect measurements, the quality of input sensor data is also noisy [21]. The errors can be easily accumulated and reflected from the derived parameters, impairing the final measurement accuracy. However, the object size model in *Aware* also needs certain user-specific profile. Enforcing its setting to end-users certainly incurs inconvenient user experience, and the configuration errors from inexperienced users could further continuously degrade *Aware*'s performance.

(3) *Crowdsourcing with vicinal objects of similar sizes.* In the crowdsourcing mode of *Aware*, objects of similar sizes may co-exist in the vicinity, and users' measured sizes also contain errors. We need to effectively identify the measured sizes that correspond to the same object before the fusion.

**Contributions.** To cope with the above challenges, we propose a series of key techniques in the *Aware* system design. The contributions can be summarized as follows:

(1) We study the geometric relation of user's upper part of the body and propose a new model that converts the indirect sensor inputs to the desired parameter values, so that measuring object size using on-board motion sensors becomes viable. The computation of this model is lightweight, which is completely affordable on wearable or mobile platforms.

(2) We remove the dominating sensor errors in deriving model parameters from the low-accuracy motion sensor inputs, such as from accelerometers and gyroscopes. We further propose refinement techniques that leverage user's historical measures and data independence to enhance the parameter and model accuracy. However, all user-dependent profiles are automatically configured without user's explicit input.

(3) To fuse the crowdsourced object size measures, we propose a density-based clustering approach to identify the measured sizes that correspond to the same object, based on which *Aware* could aggregate different measures from crowds and return the much improved crowdsourced object size result to users.

We implement *Aware* on the Android platforms. Extensive experimental studies with four users in both indoor and outdoor environments show that the object size measure error can be 8% on

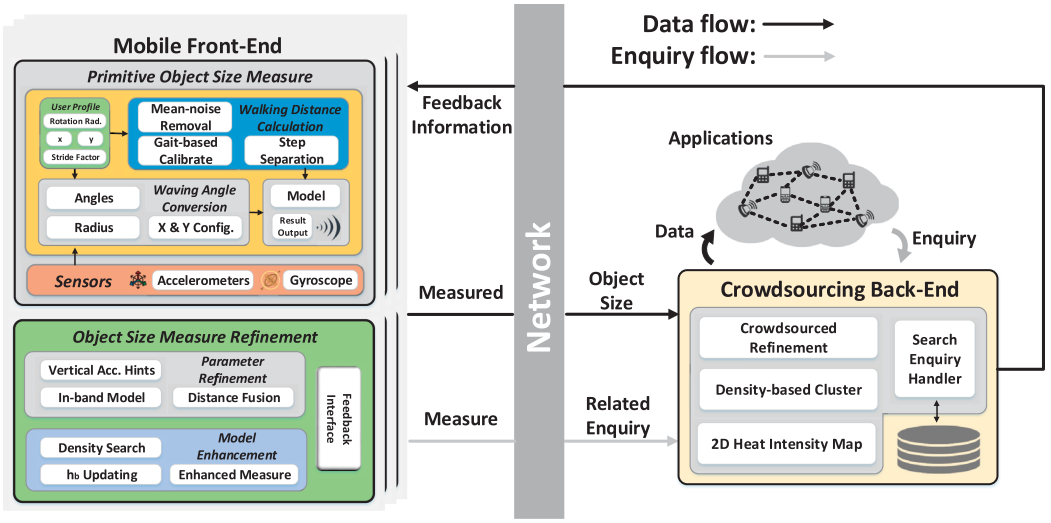


Fig. 2. **Architecture of *Aware* with mobile front-end and crowdsourcing back-end.** The mobile front-end completely fulfills the object size measure function and all computations are local on the devices. The back-end design provides an advanced crowdsourcing feature for both the performance enhancement and the result sharing.

average, which is sufficient for many useful daily applications. In the experiment, we also find that our profile automation design can provide a great convenience for *Aware* users by avoiding an explicit user parameter measure. In addition, we investigate *Aware* in the crowdsourcing mode, which further improves the performance by 78% to 22% in different scenarios, compared with the accuracy from individual measures.

**Roadmap.** The rest of this article is organized as follows. We conduct an overview in Section 2 and detail the *Aware* design from Sections 3 to 5. We implement *Aware* and evaluate its performance in Sections 6 and 7. The related work is reviewed in Section 8. We introduce the limitations of the current *Aware* design in Section 9 before we conclude in Section 10.

## 2 *AWARE* OVERVIEW

The *Aware* design strives to fulfill many daily application needs (Section 1) that require a convenient method to estimate object size, where the object can be large in size and not directly touchable by the user. *Aware* is thus positioned as *a pioneer attempt of such a contactless approach that can achieve a relatively high accuracy to satisfy the application demands, instead of the replacement of any professional precise measuring instrument.* In this section, we first introduce the system architecture, followed by the detailed designs in the following sections.

The architecture of the *Aware* service, at a high level, is composed of two primary parts: (1) the mobile front-end, and (2) the crowdsourcing back-end, as illustrated in Figure 2. The mobile front-end is a compulsory module for each *Aware* user, which completely fulfills the object size measure function. The back-end design provides a crowdsourcing feature (optional yet encouraged) for both the improvement and result sharing.

### 2.1 Mobile Front-end

In the mobile front-end, we first achieve a primitive object size measure design, and then leverage several key insights to further enhance *Aware*' performance.

**Primitive Object Size Measure (Section 3).** We investigate the geometric relation of human's upper part of body and propose a new model to derive the object size (Section 3.1). This new model in turn requires us to cope with two main tasks (Section 3.2):

*Walking distance calculation.* User's walking distance between two measurement locations is a core parameter in the object size model. However, this distance is usually short in *Aware*, e.g., a few steps, which prohibits the adoption of existing approaches (detailed in Section 3.2.1). In light of this, we analyze user's walking pattern and propose a gait-based calibration technique. It can exclude dominant sensing noises by leveraging a speed consistence feature and dramatically reduce the calculation error for user's walking distance.

*Waving angle conversion.* The object size measure model also needs user's sight-line rotation angles (with respect to the horizontal plane) during arm's waving. This component can convert indirect sensor inputs, which traces arm's movement, to the desired sight-line angular values. The conversion, however, requires certain user-dependent parameters to assist. We thus further propose to automatically configure these parameters, without user's explicit input (Section 3.2.2).

**Object Size Measure Refinement (Section 4).** With the above primitive object size measure design, we find through experiments an urgent desire for the design refinement and also observe valuable opportunities from two aspects.

*Parameter refinement.* Among all parameters used in the model, the accuracy of the user's walking distance still dominates the final performance, and we have significantly improved its accuracy. In this component, we further propose a fusion mechanism that can leverage the unused vertical motion hint from stride estimation to augment calculation's robustness and reliability (Section 4.2). The refined walking distance can also be used to recalibrate other parameters.

*Model enhancement.* This component enhances *Aware*'s object size measure model. In particular, when the bottom of a measured area is on the ground, e.g., a very common case when measuring an object's height, the model can be improved by using an alternative model expression with less input errors, and we leverage this opportunity to further improve the measurement accuracy (Section 4.3).

### 2.2 Crowdsourcing Back-end

In the crowdsourcing mode, in addition to the submission of measured object size, a user also needs to upload its own location, walking direction and relative distance to the object. Such information can be obtained along with the object size measure (Section 5). We allow users to disable the crowdsourcing feature according to their privacy preferences.

To fuse the results with crowdsourcing, the back-end engine projects individual size measures to a 2D spatial plane, generates an intensity map and leverages density-oriented clustering to identify the measured sizes that correspond to the same object, based on which we can aggregate different measures from crowds and generate crowdsourced result.

## 3 PRIMITIVE FOR OBJECT SIZE MEASURE

We introduce our proposed object size measuring model of *Aware* in this section.

### 3.1 Object Size Measure Model

To measure the object size, e.g., the tree height from $a$ to $b$ in Figure 3, a user wears or holds the device (with the *Aware* service running), stretches the arm to point at $a$, and waves the arm until it points at $b$. This process is then repeated at another location a few steps away.

As depicted in Figure 3, a horizontal plane $\alpha$, passing through user's eyes, divides user's *sight-line rotation angle* into $\theta_t$ and $\theta_b$ (or $\theta'_t$ and $\theta'_b$) two parts. The to-be measured object size $h$ is

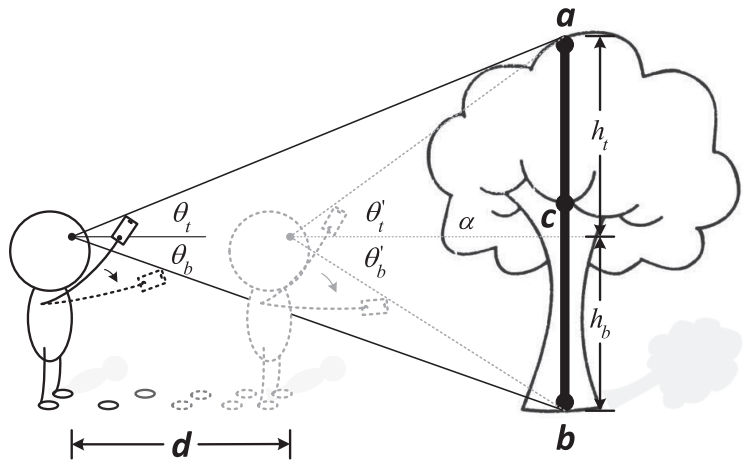


Fig. 3. **Basic object size measure principle of *Aware*** . To calculate the object size from $a$ to $b$, the model requires to derive user's walking distance $d$ and sight-line rotation angles $\theta_t$ and $\theta_b$ (as well as $\theta'_t$ and $\theta'_b$) using motion sensors.

also partitioned into two segments $h_t$ and $h_b$ by plane $\alpha$, where $h = h_t + h_b$. According to the two tangent values, $\tan(\theta_t)$ and $\tan(\theta'_t)$, we observe the difference between $h_t/\tan(\theta_t)$ and $h_t/\tan(\theta'_t)$ is equal to user's walking distance $d$, i.e., $d = h_t/\tan(\theta_t) - h_t/\tan(\theta'_t)$. Similarly, the difference between $h_b/\tan(\theta_b)$ and $h_b/\tan(\theta'_b)$ also equals to the walking distance $d$. We thus have

$$\begin{cases} h_t = d/(\frac{1}{\tan(\theta_t)} - \frac{1}{\tan(\theta'_t)}), \\ h_b = d/(\frac{1}{\tan(\theta_b)} - \frac{1}{\tan(\theta'_b)}). \end{cases} \tag{1}$$

Because $h = h_t + h_b$, object size $h$ can be calculated by

$$h = d \times \left( \frac{\tan(\theta'_t) \cdot \tan(\theta_t)}{\tan(\theta'_t) - \tan(\theta_t)} + \frac{\tan(\theta'_b) \cdot \tan(\theta_b)}{\tan(\theta'_b) - \tan(\theta_b)} \right). \tag{2}$$

To measure the object size $h$ using Equation (2), we thus need to determine the following parameters:

- User's walking distance $d$, and
- User's sight-line rotation angles, $\theta_t$, $\theta_b$, $\theta'_t$, and $\theta'_b$.

User's walking is immediately related to device's motion sensor readings (Section 3.2), while the sight-line rotation angles are not directly measurable, because wearable or mobile devices track only the attitude of user's arm, e.g., $\hat{\theta}_t$ and $\hat{\theta}_b$ with respect to the horizontal plane in Figure 4, instead of the sight-line rotation angles. To cope with this issue, we analyze the geometric relation of a user's upper part of the body. Figure 4 depicts that for sight-line angle $\theta_t$, we have an alternative expressions for $\tan(\theta_t)$, e.g., $\tan(\theta_t) = (a \times \sin(\hat{\theta}_t) - y)/(a \times \cos(\hat{\theta}_t) - x)$, where the arm rotation radius $a$ and the two user-dependent parameters $x$ and $y$ are illustrated in the figure. Similarly, we have an alternative expression for $\tan(\theta_b)$ as well. Thus, a user's sight-line rotation angles $\theta_t$ and $\theta_b$ (also $\theta'_t$ and $\theta'_b$) in Equation (2) can be calculated by

$$\theta_t = \arctan((a \times \sin(\hat{\theta}_t) - y)/(a \times \cos(\hat{\theta}_t) - x)), \tag{3}$$

$$\theta_b = \arctan((y - a \times \sin(\hat{\theta}_b))/(a \times \cos(\hat{\theta}_b) - x)). \tag{4}$$

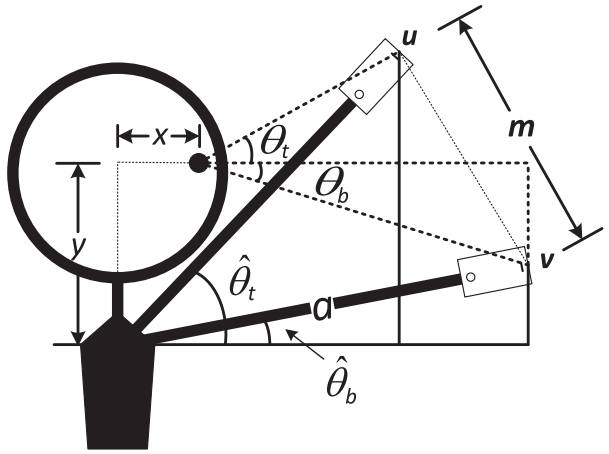


Fig. 4. **Illustration of the angle conversion from user's arm waving to the sight-line rotation**.

Table 1. Parameters in the Object Size Measure Model

| Parameters | Description |
|---|---|
| $d$ | User's walking distance |
| $\hat{\theta}_t$ and $\hat{\theta}_b$ | Arm's angular attitudes |
| $a$ | Arm rotation radius |
| $x$ and $y$ | User-dependent factors as in Figure 4 |

With Equations (3) and (4), to derive object size $h$ in Equation (2), all the parameters to be determined are listed in Table 1.

In fact, the object size model derived above can be further extended to handle the following two variant scenarios:

(1) *Bottom of measured area is not on the ground.* We note that the principle in Equation (2) does not require the bottom of the measured area touching the ground. Moreover, if the bottom position is even above the horizontal plane, e.g., position $c$ in Figure 3, angle $\theta_b$ (and also $\theta_b'$) becomes to be negative. As a consequence, $h = h_t + h_b$ is automatically changed to $h = h_t - h_b$ for Equation (2).

(2) *Measure the object width.* In this case, the user waves the arm horizontally and the sight-line rotation angle can be divided into $\theta_l$ and $\theta_r$ two parts as well, by a vertical plane along the user's facing direction. We can then apply a similar calculation as Equation (2) (e.g., by substituting $\theta_t$ and $\theta_b$ by $\theta_l$ and $\theta_r$, respectively) to measure the object width.

### 3.2 Deriving Model Parameters from Sensors

With the object size model proposed in Section 3.1, we need to determine the concrete parameters tabulated in Table 1. In this subsection, we first develop a collection of toolkits that utilize on-board sensors to preliminarily calculate these parameters. Of course, some of these initial calculations can be unreliable, which could limit the object size measure accuracy. As a result, in Section 4, we further propose key techniques to systematically enhance the overall performance.

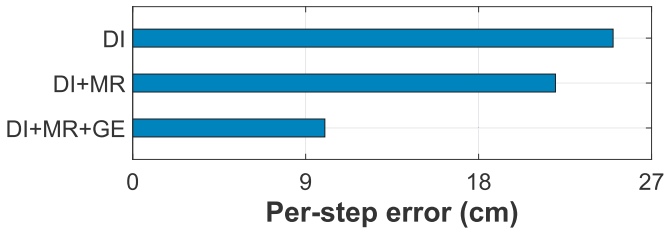


Fig. 5. **Per-step errors using three methods**. Abbreviations "DI," "MR," and "GE" stand for direct double-integral, mean-noise removal and gait-based calibration, respectively.

*3.2.1 Walking Distance $d$.* The walking distance $d$ in *Aware* is usually short, e.g., a few steps. This prohibits the adoption of GPS, as its location error about 10 to 20m [21] could overwhelm the final result. However, the calculation of $d$ needs to be completely local on the device (no ambient infrastructures required), because the user may merely use *Aware*'s mobile front-end in diverse environments. This thus excludes indoor localization techniques [32], which reply on the infrastructure support.

Traditional stride estimation techniques [1, 2, 34] could use motion sensors to estimate user's each walking step length. However, they need to manually measure the walking ground truth to train a stride estimation model in advance. The training error from inexperience users could cause continuous performance deterioration and training certainly incurs inconvenience as well.

In *Aware*, we propose a *training-free* approach using accelerometers (assisted by the gyroscope) to directly calculate the walking distance $d$, through the double-integral of the accelerations along user's walking direction. However, the direct double-integral result is known to be erroneous [34]. As in Figure 5, the per-step distance error can be up to 25cm on average with around 67cm per step.

**Mean-noise Removal.** Given $k+1$ acceleration samples $\{a_i\}$ for a device's movement, where $i = 0, 1, \ldots, k$, if its initial and final velocities, denoted as $v_0$ and $v_k$, respectively, both equal to zero, the mean-noise removal technique[1] can be applied to improve the accuracy of calculated distance. In *Aware*, both $v_0$ and $v_k$ satisfy this requirement, as the user is static before and after walking.

However, the mean-noise removal is most effective when the device's moving distance is relatively short [30], because the compensated linear acceleration noise dominates the error in this case. In Figure 5, we apply double-integral to the mean-noise removed accelerations. The result shows that the performance can be improved, but very limited.

**Gait-based Calibration.** To further prompt the accuracy, we observe that a user's walking in *Aware* is essentially composed of three phases: (1) speeding up in the first one or two steps, (2) remaining a relatively stable (average) speed in the middle,[2] and (3) slowing down in the last one or two steps. The stability of the speed in the middle phase can be further utilized to enhance the accuracy in calculating walking distance $d$.

When a user walks, steps can be identified from vertical accelerations [34], e.g., two consecutive peaks or valleys correspond to one step in Figure 6(a). Due to the speed stability feature,

[1]Mean-noise removal technique calibrates each acceleration sample $a_i$ by a same factor so that given $v_0 = 0$, the calculated $v_k$ equals to its expected value, i.e., zero.

[2]This feature exists, when a user follows his/her natural gait pattern without an intentional change in the walking. As each user in *Aware* walks a few steps merely, it is reasonable to assume they are cooperative, e.g., using their natural gait patterns.

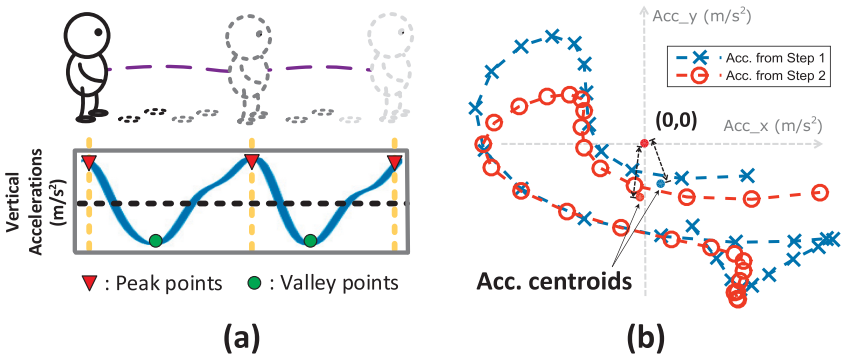


Fig. 6. **Calculating walking distance** $d$. **(a)** Each step is identified through the vertical acceleration peaks or valleys, and **(b)** offsets are observed within each middle-phase step.

the centroid of the accelerations projected on the horizontal plane within each middle-phase step should be zero, e.g., the **average** speed remains. However, as Figure 6(b) reveals, an offset can be always observed. This is mainly due to (1) non-linear acceleration errors that cannot be removed by the mean-noise removal (our target here), and (2) inevitable gait irregularity (even a user walks following its natural gait pattern with a relative average walking speed).

In *Aware*, we propose to further compensate the accelerations by enforcing the centroid of horizontal accelerations within each middle-phase step to be zero. Although the gait irregularity should distort the stability feature, the accuracy improvement achieved in Figure 5 indicates the non-linear acceleration errors dominate the offset, where the per-step error reduction is above 50%.

Although the walking distance accuracy can be dramatically improved, we find there is still room for a further refinement in Section 4.2. Therefore, according to above observations and experiments, the preliminary walking distance $d$ in *Aware* is calculated as follows:

- Apply mean-noise removal to remove the linear error for all the acceleration samples.
- For speeding up and slowing down two phases, e.g., the first step and the last step, the distance is computed by double-integral using mean-noise removed accelerations.
- For each middle-phase step, accelerations are further purified by enforcing the acceleration centroid to zero, based on which the distance is computed by the double-integral.

*3.2.2 Waving Angle Conversion.* In addition to user's walking distance $d$, *Aware*'s object size model also needs user's sight-line rotation angles, $\theta_t$, $\theta_b$, $\theta'_t$, and $\theta'_b$ in Equation (2), during arm's waving. In this subsection, we convert indirect sensor inputs, e.g., (1) arm's angular attitudes $\hat{\theta}_t$ and $\hat{\theta}_b$ that traces arm's movement, to obtain our desired angular values, assisted by (2) arm rotation radius $a$ and (3) user-dependent parameters $x$ and $y$ in Table 1.

**Arm's Angular Attitudes $\hat{\theta}_t$ and $\hat{\theta}_b$.** When arm is straightened, its angular attitudes are consistent with device's attitudes, as Figure 4 depicts. Mobile operating systems nowadays can precisely track device's attitudes in an absolute coordinate system [36]. For instance, the API on Android platform, TYPE_GAME_ROTATION_VECTOR [5], can be used to derive the *pitch*, *roll*, and *yaw* angles of a device to represent device's absolute attitude.

From Figures 7(a)–7(c), we examine the accuracy of the angles obtained from the Android platform in three typical working environments of *Aware*: outdoor, indoor, and indoor with electromagnetism interference. The angle's rotation range is $[0, \pi]$ in the experiment. From the result,

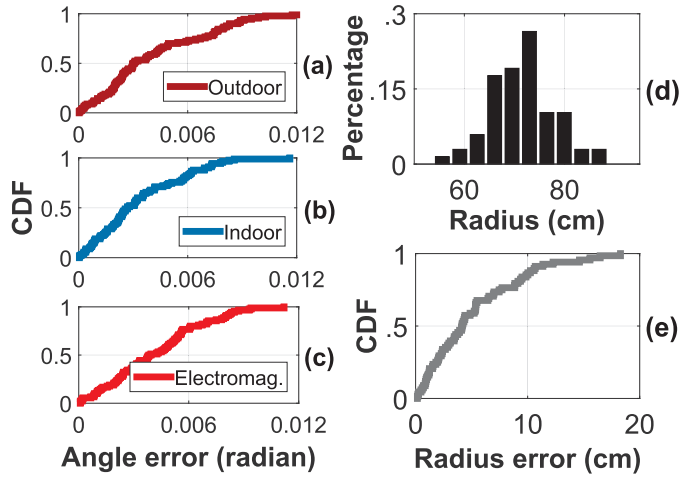


Fig. 7. **CDF of the angular and arm rotation radius errors**. Angle errors in **(a)** outdoor, **(b)** indoor, and **(c)** indoor with electromagnetism interference three environments. **(d)** Distribution of calculated arm rotation radius $a$, and **(e)** error CDF.

we observe that majority of angular errors are less than 0.01 radian, indicating that the device's angular rotations can be precisely and robustly tracked.

**Arm Rotation Radius *a*.** To automate this configuration, we can find when user's arm is waved, e.g., from positions $u$ to $v$ in Figure 4, an isosceles triangle is formed by the arm and device's displacement $m$. Hence, we have

$$a = m \Big/ \left(2 \times sin\left(\frac{\hat{\theta}_t - \hat{\theta}_b}{2}\right)\right), \tag{5}$$

where arm's angular attitudes $\hat{\theta}_t$ and $\hat{\theta}_b$ are obtained already and the displacement $m$ can be computed by the double-integral of accelerations with the mean-noise removal. Because $m$ is short, the mean-noise removal is effective [30]. In Figure 7(d), a user waves the arm multiple rounds, and we observe that the computed $a$ approximately follows a Gaussian distribution. We can thus average the calculated $a$ from multiple object size measures to approximate its true value. Figure 7(e) shows that the final error of calculated $a$ is 5.2cm on average. How such an accuracy attributes to the final object size measure performance is evaluated in Section 7.

**User-dependent Parameters $x$ and $y$.** These two parameters describe the lengths of two line segments inside human's body. Therefore, they are non-trivial to be explicitly measured. Our key idea to determine $x$ and $y$ is to treat them as variables and leverage the geometric relation to solve their values. We find that when a user uses *Aware* for the first time, if the user waves the arm to measure the same object at *three* different locations as in Figure 8, we are able to solve $x$ and $y$ directly, which is an one-time effort.

For $\theta_b$ in Figure 8, $\tan(\theta_b) = h_b/(d + d' + l')$, where the variables $h_b$, $d$, $d'$ and $l'$ are annotated in the figure. Equation (4), however, indicates that an alternative expression of $\tan(\theta_b)$ is $\tan(\theta_b) = (y - a \times \sin(\hat{\theta}_b))/(a \times \cos(\hat{\theta}_b) - x)$. Therefore, we have

$$\frac{h_b}{d + d' + l'} = (y - a \times \sin(\hat{\theta}_b))/(a \times \cos(\hat{\theta}_b) - x). \tag{6}$$

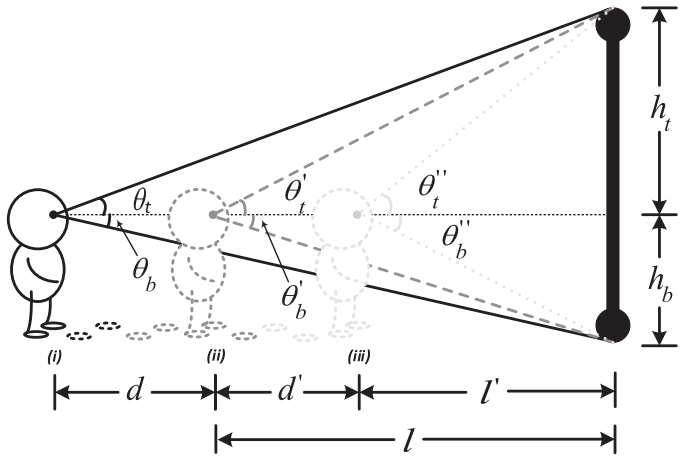


Fig. 8. **Initial configuration for parameters $x$ and $y$.**

Similar expressions can be written for other five angles, $\theta_t$, $\theta_b'$, $\theta_t'$, $\theta_b''$, and $\theta_t''$ as well, and these equations together can be rephrased as one binary quadratic function group:

$$\begin{cases} A_1y^2 + A_2yx + A_3y + A_4x + A_5 = 0, \\ B_1y^2 + B_2yx + B_3y + B_4x + B_5 = 0, \end{cases} \tag{7}$$

where $A_i$ and $B_i$, $i = 1, 2, \ldots, 5$, are composed of the following factors obtained already: (1) arm's angular attitudes: $\hat{\theta}_b$, $\hat{\theta}_t$, $\hat{\theta}_b'$, $\hat{\theta}_t'$, $\hat{\theta}_b''$, $\hat{\theta}_t''$, (2) arm's rotation radius $a$, and (3) user's walking distances $d$ and $d''$. After solving Equation (7), we can achieve the initial configuration of $x$ and $y$.

## 4 OBJECT SIZE MEASURE REFINEMENT

So far, we have shown that the object size can be measured using a group of concrete parameters through our object size measure model, and these parameters can be gained using on-board sensors.

### 4.1 Need for Refinement

Among all the parameters listed in Table 1, arm's angular attitudes $\hat{\theta}_t$ and $\hat{\theta}_b$ are reliably obtained, as Figures 7(a)–7(c) unveil.

Other three parameters are directly or indirectly obtained from accelerometers, less reliable compared with the angular values. For arm's rotation radius $a$, as device's displacement is short, the mean-noise removal is effective in this case and the accuracy can be improved by average, as Figures 7(d) and 7(e) show. In Section 7, we find that radius $a$'s error causes 2% object size inaccuracy merely.

For walking distance $d$, although we have significantly improved its accuracy in Section 3.2.1, the performance is still not precise enough (Figure 5). However, the initialization of parameters $x$ and $y$ takes $d$ as input, impacted by $d$'s accuracy as well. In Section 7, we find that the measured object size is mainly affected by the quality of $d$, e.g., 12% average performance loss. Hence, we aim to further refine the calculation of user's walking distance $d$, as well as parameters $x$ and $y$ (Section 4.2). Moreover, we also upgrade the object size model to further enhance *Aware*'s performance (Section 4.3).

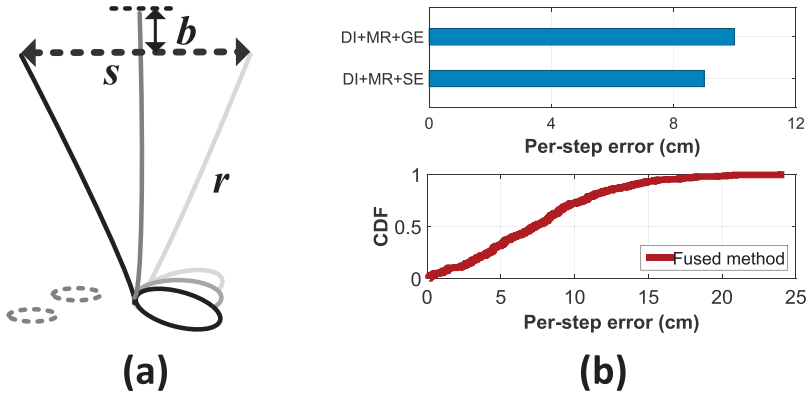


Fig. 9. **Parameter refinement**. **(a)** Stride estimation (SE) principle. **(b)** Per-step errors for gait-based calibration (DI+MR+GE) and stride estimation (DI+MR+SE), and the fusion.

### 4.2 Parameter Refinement

The calculation of the walking distance $d$ in Section 3.2.1 utilizes horizontal accelerations through double-integral, remedied by mean-noise removal and gait-based calibration. To further improve its reliability, we propose to further collaborate with existing stride estimation techniques [1, 2, 34], which mainly adopt vertical accelerations, independent with $d$'s calculation in Section 3.2.1 by purely the horizontal acceleration sensor readings.

As aforementioned, stride estimation builds a pedestrian model that requires the walking ground truth as a prior for training. Hence, we propose to utilize the walking distance $d$ derived from Section 3.2.1 as benchmark to train the stride estimation model, and the model can be gradually updated further when the user measures more objects (with more computed $d$s) until it becomes stable. The distance $d$ is finally from their fusion. This combination can, thus:

- augment $d$'s accuracy further, as it is less likely that two methods are concurrently unreliable,
- and avoid explicit model training for stride estimation.

**In-band Stride Model Building.** The stride estimation model first derives user body's vertical displacement (*a.k.a.* bounce $b$) within each step using accelerometer readings [34], as depicted in Figure 9(a). Bounce $b$, together with user's leg length $r$, could form the equation $r^2 = (s/2)^2 + (r - b)^2$, where $s$ is the length of one step. According to Reference [34], we have the following relation:

$$\begin{aligned} s &= k \cdot \sqrt{r^2 - (r-b)^2} \\ &= k \cdot \sqrt{2br - b^2} \approx k \cdot \sqrt{2br} = k' \cdot \sqrt{b}, \end{aligned} \tag{8}$$

where $k$ represents a calibration factor, $k'$ emerges user's leg length $r$ and $k$ as a single factor, e.g., $k' = 4kr^2$, and the approximation is because $b \ll r$. Thus, factor $k'$ needs to be trained in advance based on a set of measured walking distances.

After a user measures $n$ object sizes, *Aware* gets $\{d_i\}$, where $i = 1, 2, \ldots, n$ and each $d_i$ is calculated from Section 3.2.1. If we denote $d_i = \sum_{j=1}^{m_i} s_{i,j}$, where $s_{i,j}$ is the length of the $j$th step and $m_i$ is the number of steps within $d_i$ (e.g., $m_i$ can be counted by the number of peaks from vertical

accelerations as in Figure 6), then factor $k'$ in Equation (8) can be determined by

$$k' = \arg_{k'} \min \left\{ \sum_{i=1}^{n} \left\| \frac{1}{m_i} \left( d_i - \sum_{j=1}^{m_i} s_{i,j} \right) \right\|_2 \right\},$$
$$= \arg_{k'} \min \left\{ \sum_{i=1}^{n} \left\| \frac{1}{m_i} \left( d_i - \sum_{j=1}^{m_i} k' \cdot \sqrt{b_{i,j}} \right) \right\|_2 \right\}, \quad (9)$$

where $b_{i,j}$ is the bounce for the $j$th step within $d_i$. Note that $k'$ can be obtained when *Aware* is used for the first time, e.g., $n = 1$, and it can be gradually updated until its value becomes stable. Through the experiment, we observe that initially $k'$ may fluctuate. After it is refined for five to eight times, $k'$ could become stable. Therefore, this model refinement design is efficient in practice.

**Walking Distance Fusion.** For gait-based calibration (Section 3.2.1) and stride estimation, *Aware* maintains their average step lengths, denoted as $\bar{s}_g$ and $\bar{s}_t$, respectively. To measure one object, after user's walking, *Aware* first calculates the distance using two methods independently, e.g., $d_g$ and $d_t$. If the resulting per-step lengths are both greater or less than their respective average value, then we call it *consistent*; otherwise, it is *inconsistent*. The fused $d_f$ is determined:

$$d_f = \begin{cases} (d_g + d_t)/2, & \text{if consistent,} \\ (\bar{s}_g + \bar{s}_t) \times m/2, & \text{if inconsistent,} \end{cases} \quad (10)$$

where $m$ is the number of steps during walking. Figure 9(b) depicts the per-step length error of two individual approaches are 10cm and 9cm, respectively, and the fused method can further reduce the error below 7cm on average.

After the stride estimation model becomes stable, we can apply Equation (10) to the acceleration readings in the initial configuration of parameters $x$ and $y$ (Figure 8) to further refine their values.

### 4.3 Model Refinement

We propose to further enhance the object size model in this subsection by leveraging the following observation. When the bottom of the measured area is on the ground (we name it *contacting-ground measure*, e.g., the most common case to measure an object's height as in Figure 8), the obtained $h_b$ should always be identical even a user measures different objects, as $h_b$ in this case describes the height of user's eyes with respect to the ground. Thus, for a contacting-ground measure, its size $h = h_t + h_b$ can be alternatively derived from

$$h = l \times \tan(\theta'_t) + h_b = \frac{h_b}{\tan(\theta'_b)} \times \tan(\theta'_t) + h_b, \quad (11)$$

where $\theta'_t$ and $\theta'_b$ are user's sight-line angles at the second measurement position, e.g., position (*ii*) in Figure 8 that is separated by $d$ from the first position (*i*), and $l$ is the distance between position (*ii*) and the object.

Equation (11) does not directly adopt $d$ (e.g., dominating error source to the final measured $h$, even we have significantly reduced $d$'s error). If we can leverage the identical attribute from the contacting-ground measures to purify $h_b$ with a high precision, then Equation (11) can further improve $h$'s accuracy.

**Purifying $h_b$.** Given one object size measure, *Aware*, however, is not aware whether the bottom of the measure area is on the ground or not, and we do not expect users could explicitly provide such feedback. To first identify such cases, we observe that during the measurement, if both $\hat{\theta}_b$ and $\hat{\theta}'_b$ are positive, i.e., user arm's waving stops below the horizontal plane, then it is *highly likely*

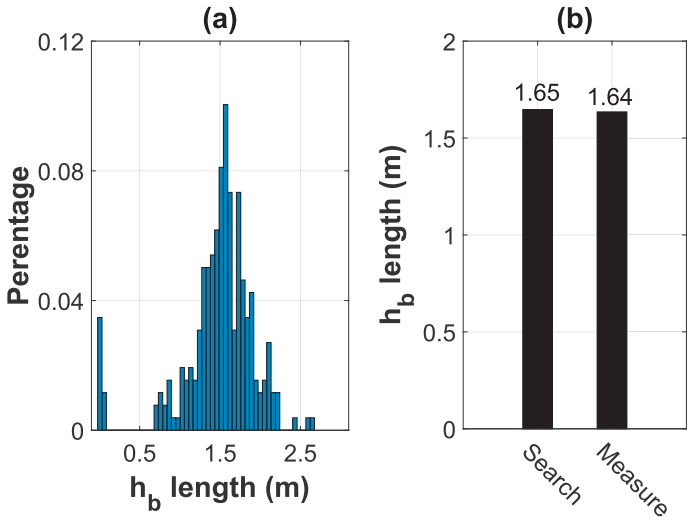


Fig. 10. **Model refinement**. **(a)** Distribution of $h_b$ within the candidate set, and **(b)** the searched $h_b$ compared with the measured result.

a contacting-ground measure. Of course, this is not always correct and the selection may include some undesired measures, e.g., the bottoms of the measured areas are close but not on the ground. Nevertheless, we find that this rule could serve as a coarse filter to select a candidate set, in which the majority is our desired measures.

Within this candidate set, the $h_b$ from each contacting-ground measure should be centered around the true $h_b$ value. The $h_b$ from other undesired measures will be scattered. If we simply average all $h_b$s from the candidate set to approximate $h_b$'s true value, then the result will be drifted by these undesired measures. To solve this issue, we observe that if we project all the $h_b$ values from the candidate set to an axis, the data density around $h_b$'s true value will be very high. Therefore, we use a sliding window to compute the density of $h_b$'s distribution. The position that leads to the highest density estimates the true $h_b$ value, as in Figure 10(a). As a matter of fact, $h_b$'s error mainly comes from user's walking distance $d$ as well. This refinement essentially leverages the collective distribution of $h_b$ to mutually cancel out their non-linear (Gaussian) errors, e.g., the estimated $h_b$ is very close to the measured $h_b$ value as in Figure 10(b). The refined $h_b$ thus has much higher precision compared with the walking distance $d$ derived from each individual object size measure. As the result, Equation (11) can dramatically enhance the performance of the final result.

Now with the high quality $h_b$, for a new measure, if it is classified into the candidate set, its $h$ can be calculated using Equation (11). In Section 7, we find even for an undesired measure within the candidate set, the calculated $h$ using Equation (11) improves. This is because the error in $d$ overwhelms the $h_b$'s difference between the undesired and desired measures within the candidate set.

## 5 CROWDSOURCING BACK-END

With the object size model design in Sections 3 and 4, we further introduce *Aware*'s crowdsourcing back-end in this section. The crowdsourcing engine could fuse individually measured object sizes and provide users the crowdsourced result.

**Geographic Partitioning.** To fuse measurements from different users, we should first group these size measures that correspond to the same object. Therefore, we need each object's location on the

**ALGORITHM 1:** Density-based clustering

```
1  input: a set of object sizes h = {h_i};
2  output: the set p = {p_j}, where each p_j is the h value of one density peak;
3  h_min = min_{h_i}{h};
4  h_max = max_{h_i}{h};
5  for search h in [h_min, h_max] do
6      for search w in [0, w_max] do
7          D[h] = D[h] + f(h, w);
8  Conduct curve fitting to D[h];
9  Record the h value for each local maximum peak on the fitted curve as p = {p_j};
10 return p;
```

2D plane at the first place. Although such location information is not directly available, it can be calculated from:

- User's location, e.g., position (*ii*) in Figure 8 obtained from GPS or indoor localization [32].
- Walking direction, e.g., obtained from hardware or software compass [23, 36].
- Distance between the user and object, e.g., $l$ in Figure 8 that can be computed by $l = h_t / \tan(\theta_t')$.

If a user has a privacy concern for uploading above information, then the crowdsourcing function can be disabled. Due to sensor noises, e.g., GPS, compass, and so on, and measurement errors, e.g., $l$, we can only infer one possible range about object's true location based on the information above. With more ranges obtained from different users, we can gradually shrink its spatial uncertainty. To this end, we adopt the method in Reference [21] to represent the possible locations for each object as a trapezoid shaped polygon. All polygons are then projected to the 2D plane to generate a spatial spectrum, where the intensity of each point on the 2D plane indicates the amount of overlapped polygons. We can then search for connected spectrum components and each component contains a vicinity of measured objects [21].

Of course, different objects can be classified into the same connected component, e.g., their locations are close. We need to further distinguish them before the object size fusion.

**Density-based Clustering.** For every connected component, we can apply the similar observation as in the model refinement (Section 4.3). If different users measure the same object, then the measured $h$ values should form a cluster. We can thus use a sliding window to conduct the data value density search, and the density peak approximates to the true $h$ value.

Because one connected component may contain different objects with similar sizes, multiple peaks can appear. One fixed sliding window size $w$, however, cannot fit all different components, as the sizes and numbers of objects in each component can be different. To address this issue, we treat window size $w$ as a variable and propose a joint search algorithm applied to both object size $h$ and $w$ in Algorithm 1. For each connected component, we first project all its contained $h$ values to an axis. We search from the minimum to maximum values of $h$, and examine the number of density peaks to be discovered. For each possible object size value $h$, we accumulate the data value density, e.g., $f(h, w)$ in line 7, by varying $w$ to a sufficiently large sliding window size value, e.g., $w_{max}$ is empirically set as 20cm in Section 7. If the current $h$ really represents the size of one object, then the accumulated result always indicates so; otherwise, they cannot form such a local maximum value. Figure 11 shows an example of Algorithm 1.

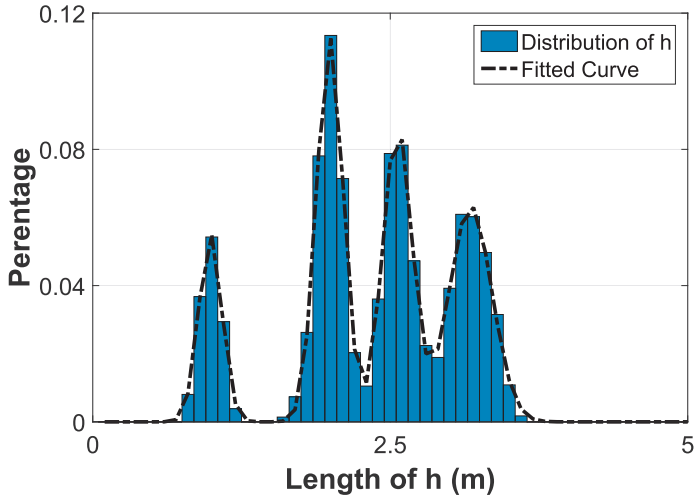


Fig. 11. **Density-based clustering example.** The searched $h$ values for four objects by Algorithm 1 are 1.0m, 2.0m, 2.6m, and 3.1m, respectively, very close to the measured ground truths, which are 0.96m, 2.0m, 2.8m, and 3.2m, respectively.

**Object Size Inquiry Handling.** If a user enables the crowdsourcing function, then after each object size measure, the measured object size, together with user's location and direction information, will be submitted to the crowdsourcing back-end. For each submission, if its polygon does not intersect with any existing spectrum component, a new component is formed and the fused result is just this newly submitted one; otherwise, the polygon overlaps with one existing spectrum component. Now for this component, we launch Algorithm 1 to update all density peaks and classify the submitted object size to the closest peak. The $h$ value of this peak is thus the fused result, returned to the user.

## 6 IMPLEMENTATION AND SETUP

We develop an *Aware* system on Android platforms and experiment with LG watch and HTC phone.

**Mobile Front-end.** In the mobile front-end, we use TYPE_ACCELEROMETER and TYPE_GAME_ROTATION_VECTOR two APIs to acquire three-axis accelerations and the *pitch*, *roll*, and *yaw* angles to represent device's absolute attitude, respectively. For the walking distance calculation, we implement both our gait-based calibration scheme and the state-the-art stride estimation model [34], as well as the in-band stride estimation model building and the distance fusion. To configure user-dependent parameters $x$ and $y$, a user waves the arm to measure one object at three different locations when *Aware* is launched for the first time. After the initial parameter setting, when objects are measured, *Aware* further detects whether they are contacting-ground measures, e.g., the bottom of the measured area is on the ground. If so, then *Aware* is able to update the recorded $h_b$ and enhance the result using the refined object size model.

**Crowdsourcing Back-end.** The crowdsourcing back-end is an advanced feature in *Aware*. We implement the density-based clustering algorithm that can distinguish nearby objects of similar object sizes and further fuse the measured sizes belonging to the same object to further enhance the measurement accuracy. In this algorithm, we empirically configure the upper bound of the

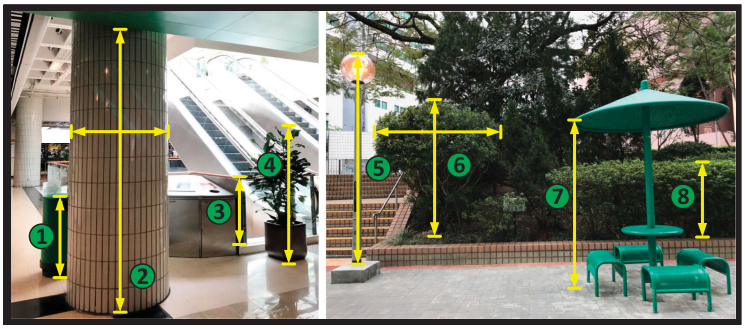


Fig. 12. **Eight sample objects measured by *Aware* in the experiment for indoor (left) and outdoor (right) environments.**

sliding window size search range as 20cm. Under the crowdsourcing model, the performance is evaluated using the crowdsourced result.

**Experiment Methodology.** With the experiment setup above, we evaluate *Aware*'s performance in both indoor and outdoor environments with four users, i.e., 1 female and 3 males with body heights from 1.68m to 1.85m. We use *Aware* to measure 40 objects with the ground truth varying from .65m to 4.25m and each object is measured 6 to 10 times. The distance between each object and user varies from 2.3m to 9.3m during the measurement. Among the 40 measured objects, there are 30 height measures (24 contacting-ground cases and 6 hanging in the area cases) and 10 width measures. Figure 12 depicts several objects that are measured in the experiment. In the experiment, we also compare *Aware* with a camera-based method CamMeasure [3], a commercial APP in the Apple Store. There are some other APPs of this kind in APP stores but they share a similar design principle, so we do not compare *Aware* with them individually.

## 7 PERFORMANCE EVALUATION

**Accuracy.** We first examine *Aware*'s end-to-end object size measuring accuracy and plot its per-unit measurement error. For the object height measure, Figure 13(a) shows that the average error is 8% and 80th percentile error is 12%. For the object width measure, the average and 80th percentile errors are 14% and 23%, respectively (slightly increased). The object height measure outperforms the width measure mainly benefits from the model refinement design (Section 4.3), so that we can use the refined model in Equation (11) to directly derive object height $h$ without using the derived on-time walking distance. Figure 13(a) indicates that *Aware* overall achieves relatively high accuracy.

**Impacts from Model Parameters.** For all four sets of parameters in our object size model, Figures 7(a) to 7(c) has shown the angular accuracy is very high and reliable. In Figure 13(b), we further examine the rest three sets. We first provide the ground truth for the rest parameters and replace one of them with the calculated value each time. Figure 13(b) shows that "$d$," "$a$," and "$x$ and $y$" could cause 12%, 2%, and 2% accuracy reduction, respectively. The results indicate that $d$ is dominating in deriving the object size than other two parameters. As a result, the error from $d$ impacts more on the derived object size than other two parameters, which motivates us to further enhance its accuracy in Section 4.

**Performance Gain from Refinements.** In Sections 3 and 4, we first introduce a primary version of *Aware* and then enhance it by parameter and model refinements. We investigate performance gains from such a design in Figure 14. From the result, we see that compared with the basic

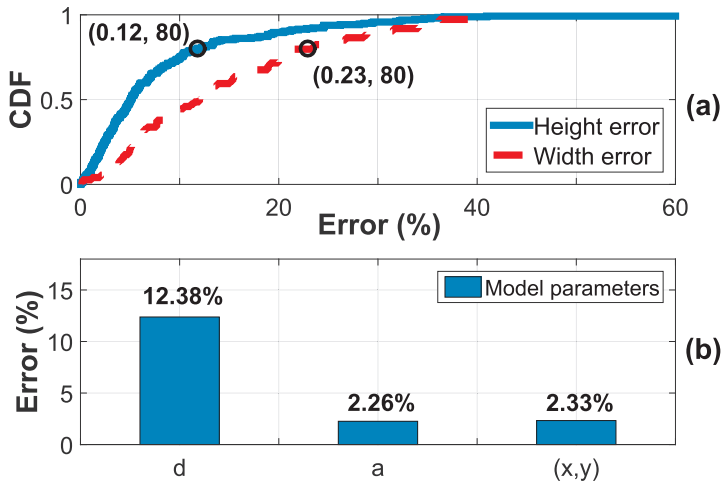


Fig. 13. ***Aware* 's overall performance.** (**a**) Height and width measures, and (**b**) the impacts from different model parameters.

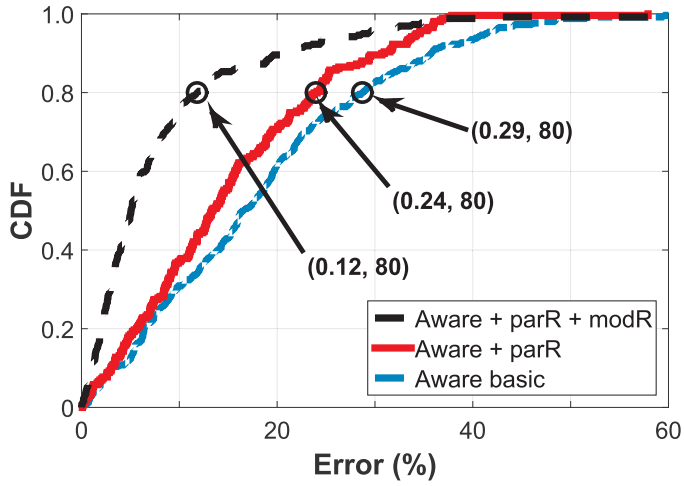


Fig. 14. **Performance from different versions of *Aware*: basic, basic plus parameter refine, and basic plus parameter and model refine.**

version (*Aware* basic), the parameter refinement (*Aware* + parR) improves the accuracy of walking distance $d$, leading to 3% average error reduction in the object size measure. As the refined model (*Aware* + parR + modR) does not directly utilizes user's walking distance $d$, the final performance is further improved by 6.8%, e.g., 9.8% average error reduction in total. Figure 14 indicates that our enhancement designs are effective, where the parameter and model refinements contribute to 31% and 69% size measure error reductions, respectively.

**Performance Comparison.** In this experiment, we compare *Aware* with CamMeasure [3] that uses camera to measure the object size. Figure 15(b) shows that for contacting-ground cases, *Aware* can achieve comparable performance as CamMeasure, while CamMeasure works poorly for non-contacting-ground cases as in Figure 15(a). We notice that this is a methodology limitation that

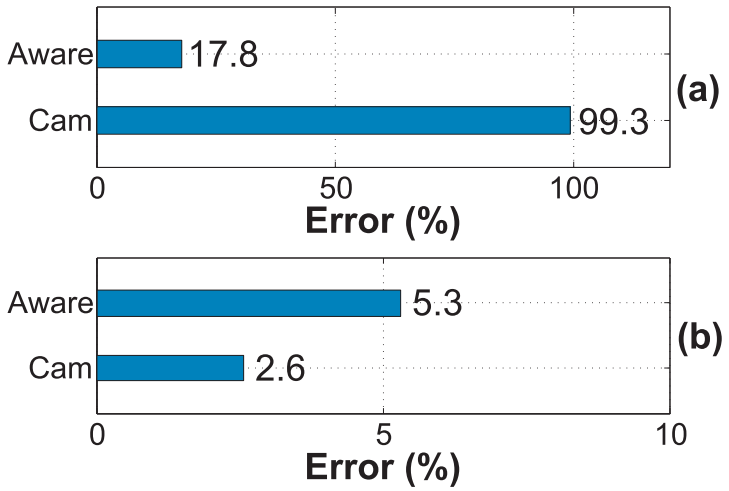


Fig. 15. **Comparison with CamMeasure that uses the camera for measurements.** (**a**) Non-contacting and (**b**) contacting-ground cases.

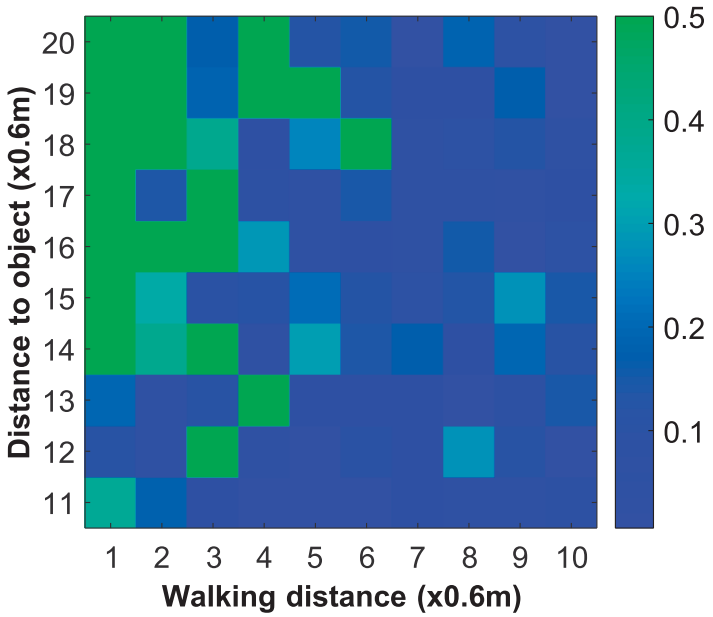


Fig. 16. **Object size measure impacted by the user-object distance and walking distance.**

exists in a variety of similar approaches as CamMeasure in the APP store. Since *Aware* has no special requirements for the to-be measured object and hardware, e.g., camera is unavailable on wearables, it serves as a more general solution to fulfill the application needs in practice.

**Impacts of Walking Distance and User-object Lengths.** In this trial, we adopt 0.6m as an unit, e.g., the ground is covered by tiles and the tile length is 0.6m, and experiment with the object of size 2.75m by varying walking distance in Figure 16, where the color density means errors, e.g., a deep density indicates a small error. We find when user's walking distance is sufficient long,

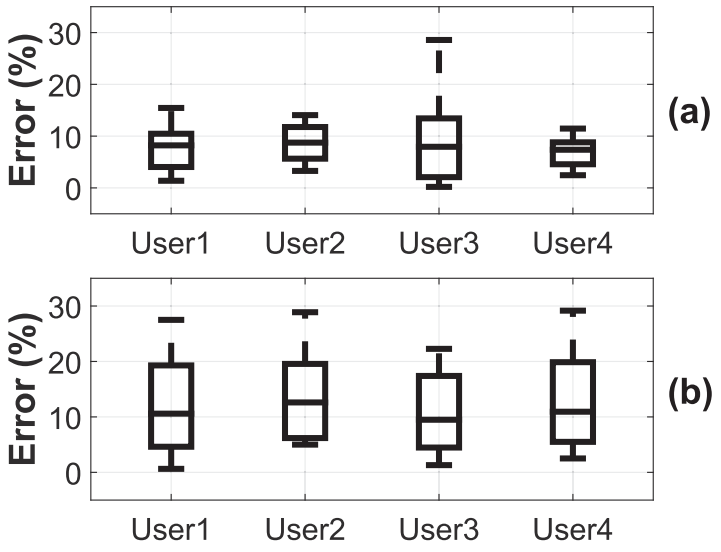


Fig. 17. ***Aware*'s performance from different users.** (**a**) Height and (**b**) width measures.

e.g., more than five tiles, the object size measure error is small in general. This is because user's sight-line rotation difference between two measuring positions becomes larger, e.g., providing better signal-to-noise ratios against the operational uncertainty. However, in practice, there is not always enough space to walk for a long distance to measure object size. Figure 16 suggests the walking distance length within 5 to 9 steps is sufficient in common cases.

However, a smaller user-object distance certainly improves the measuring results, because in this case the object is relatively large in user's eyes and it is more reliable for users to point at the boundary of the measure area on the object. According to the experiment, the *Aware* is most effective when the ratio between the user-object distance and object size is less than $13 \times 0.6/2.75 = 2.8$.

**Performance of Different Users.** In Figure 17, we further plot the performance of *Aware* achieved on different users with body heights from 1.68 to 1.85m. From the result, we can see *Aware* has stable performances across different users (around 8% and 12% on average for height and width measures, respectively). The 75th percentile errors are approximately 11% and 20% across different users for height and width measures, respectively. These results indicate *Aware* can achieve stable and satisfactory performance across different body sizes.

**Execution Time.** We examine the execution time of *Aware* on both smart watch and smart phone. The computational complexity of *Aware* is lightweight, which can be comfortably afforded by these two types of platforms. The average execution time to calculate the object size on smart watch is 0.34ms and it occasionally climbs up to 0.88ms. On smart phone, the average and maximum execution time is reduced to 0.22ms and 0.48ms, respectively.

**Crowdsourcing Performance.** In this trial, we first evaluate the density-based clustering (Algorithm 1 on p. 15), followed by the object size measure accuracy with crowdsourcing.

In Figure 18, we vary the number of objects (with similar but different sizes) in the same vicinity. When the objects are few, the searched peaks by Algorithm 1 closely match their ground truth values. When the object number increases, the searched peaks may drift due to other objects with similar sizes. Figure 18 implies that when more objects co-exist in the vicinity, some additional labels, e.g., user's textual or audio annotations, can be further adopted to jointly distinguish

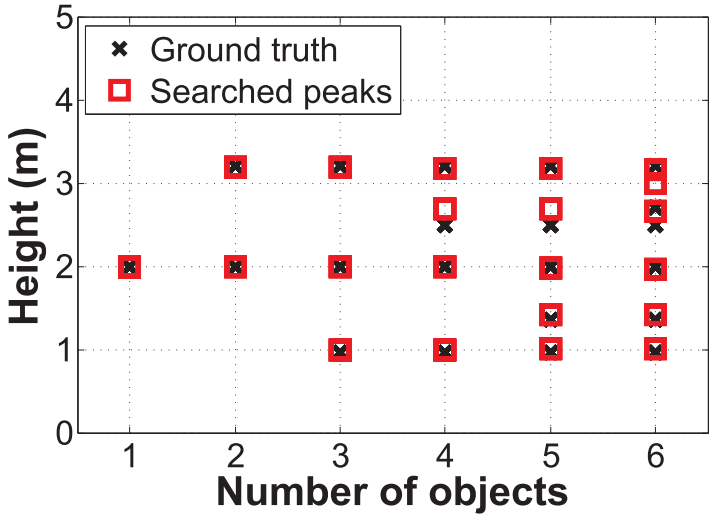


Fig. 18. **Evaluation on the density-based clustering algorithm.**

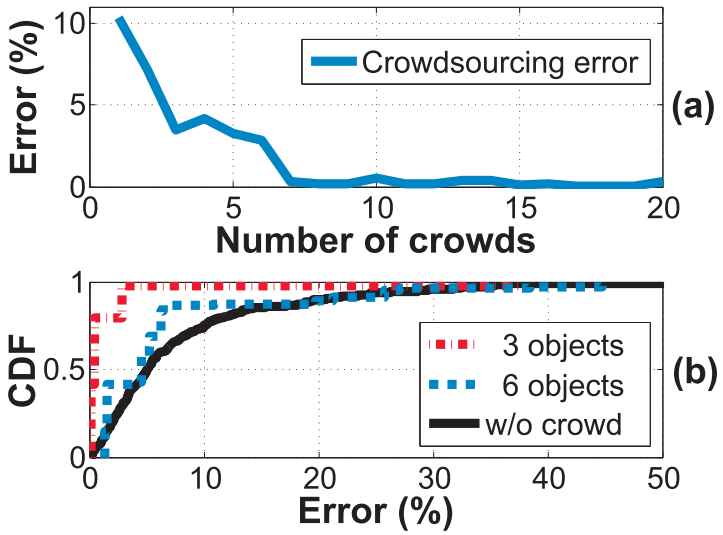


Fig. 19. **Crowdsourcing accuracy by varying the number of (a) crowds and (b) objects.**

different objects for improving the clustering performance, instead of using their sizes merely. Such an opportunity has been explored in AR services [10], and we plan to leverage it in the future.

In Figure 19, we investigate *Aware*'s end-to-end performance with crowdsourcing. Figure 19(a) first shows that after sufficient crowds contribute results for the same object (without clustering errors), the accuracy is remarkably improved, e.g., 2% error. In Figure 19(b), when objects of similar sizes co-exist in very close vicinity and the object number is small, e.g., 3, the average error per meter is 2% merely, which improves the performance without crowdsourcing ("without crowd") by 78%. Even when the object number is increased to 6, due to the peak drift in Algorithm 1, the accuracy decreases to 7%, which still outperforms "without crowd" by 22%. The result shows that

crowdsourcing is indeed capable to augment the object size accuracy, motivating users to enable this function and share their measured results.

## 8 RELATED WORK

We review the following works in the literature that are related to the design of *Aware*.

**Contactless Object Size Measuring.** Computer vision can measure the object size from multiple images pictured at multiple places in a contactless manner [6, 17]. OPS [17] is proposed to measure the distance between the user and object, which can be extended for the object's height measure. OPS, however, requires the desktop-type CPU capability, not afforded by wearable and mobile platforms. The computation overhead of other recent works, like Reference [6], can be even higher, as they demand hundreds of images about the same object and utilize expensive computer vision algorithms in the design. Although some methods [3, 19] can run on mobile devices locally, they suffer the limitations that the measured area must touch on the ground, as well as the difficulty of calibration. However, computer vision approaches [8, 19] naturally require the device equipped with the camera (unavailable on wearables) and mainly work in the daytime. To overcome above limitations, we design *Aware* by purely using motion sensors, which are widely available on a variety of wearable and mobile platforms. So far as we know, *Aware* is the first system of this kind in the community, positioned as a convenient solution to estimate the object size to fulfill a variety of application needs.

**Walking Distance Calculation.** In *Aware*, user's walking distance is usually short, e.g., a few steps. This unique feature prohibits the adoption of GPS, as its location error about 10 to 20 meters [20, 21] could overwhelm the final result. Although state-of-the-art localization techniques have significantly advanced the positioning accuracy [33], they are not suitable for *Aware*, as they reply on the infrastructure support, including RFID readers [7, 25, 29], Wi-Fi access points [12, 27, 32, 35], visible light beacons [13–15], and so on. However, users may merely use *Aware*'s mobile front-end in diverse environments, the walking distance calculation needs to be completely local on the device—no ambient infrastructures required.

Traditional stride estimation techniques [1, 2, 9, 34] could use motion sensors to estimate the length of a user's each walking step, and their summation leads to the total walking distance. However, they need to manually measure the walking ground truth to train a stride estimation model in advance. Our design in *Aware*, leveraging user's gait consistence feature to calibrate the double-integral result, is compatible to these existing techniques, which provides them an in-band stride model building opportunity. In addition, we propose a fusion design (between our gait-based calibration scheme and the state-of-the-art design [34]) to augment the walking distance accuracy further, as they conduct independent calculations using different sensor readings (horizontal vs. vertical accelerations). It is less likely two methods are concurrently unreliable.

**Motion Sensor Processing.** We mainly review accelerometers and gyroscope two types of motion sensors related to the *Aware* design. Accelerometers have been adopted in many mobile applications. The preliminary usage is to treat accelerometer readings as a mobility hint or indicator, e.g., counting steps, behavior detection [22], transport model classification [24], and so on. Some studies later leverage accelerometers to calculate the moving distance of the device through double-integral. However, the result is erroneous [34]. Recent work [30] finds that when device's displacement is short, the accuracy can be dramatically improved by the mean-noise removal. Gyroscope is widely used for orientation estimation in robotics, by leveraging the Kalman filter [18], error propagation model [11], data fusion [16]. $A^3$ [36] is later proposed for mobile platforms to achieve accurate phone attitude detection. Recently, mobile operating systems even provide

in-built APIs [5] for developers to precisely track device's attitudes in an absolute coordinate system. In *Aware*, we leverage above technical advances. For accelerometers, as the mean-noise removal technique is most effective only when the device's moving trajectory is short, we thus further propose the gait-based calibration scheme and a fusion design in the walking distance calculation.

**Crowdsourcing Applications.** The *Aware* design is orthogonal to many third-party crowdsourcing services [4, 10, 26, 28]. The object size database maintained by *Aware* can be used to enrich their data type by providing them the meta object size information. In *Aware*'s crowdsourcing back-end, we leverage the spatial partitioning method in Reference [21] to identify nearby objects, based on which we further propose a density-based algorithm to distinguish different objects of similar sizes.

## 9 LIMITATIONS AND DISCUSSIONS

We summarize the requirements and limitations of the current *Aware* design in this section.

**(1) Orientation of the mobile device:** According to the current *Aware* design, the user needs to stretch her arm during the waving, so that the device's body ($y$-axis) is along the user's arm (instead of being in an arbitrary orientation) as depicted in Figure 4, and then looks at a fixed point on the device, e.g., the middle position of the phone's boundary in the measurement. This requirement aims to obtain the angle of $\hat{\theta}_t$ or $\hat{\theta}_b$, which is needed in deriving the object size by *Aware*.

**(2) Attitude of the user's head:** In *Aware*, we need the user's cooperation not to raise or bow her head during the object size measurement. Since *Aware* calculates $\theta_t$ and $\theta_b$ in Figure 4 for deriving the object size, the raising or bowing of the user's head will thus impact the system performance. This requirement could cause certain inconvenience to the user, and we plan to study the solution when the user's head is in a more nature attitude as the future work of this article.

**(3) Walking distance measurement:** When a user walks, the actual trajectory of the user's body may not be a straight line; it is usually a zigzag pattern. If we calculate the length of each zigzag side and then add them together, then the result becomes less accurate. Therefore, in *Aware*, we compute the displacement from the starting point to the end point of the user's walking, which aligns with the user's actual walking distance. However, we need the user's cooperation to hold her phone without introducing an extra movement with respect to the user's body (e.g., the hand that holds the phone can lean on the body) during the user's walking.

## 10 CONCLUSION

This article presents *Aware*, which addresses three major challenges to turn our wearable or mobile device into a virtual ruler using motion sensors. The *Aware* design could enable a rich set of applications that require a convenient and accurate way to measure object size. We implement a prototype system on Android platforms. Extensive experiments with four users show that the error of the object size measure by *Aware* can be 8% on average and crowdsourcing can further improve the accuracy by 78% to 22% in different scenarios.

## REFERENCES

[1] Moustafa Alzantot and Moustafa Youssef. 2012. UPTIME: Ubiquitous pedestrian tracking using mobile phones. In *Proceedings of the IEEE WCNC*.

[2] Inge Bylemans, Maarten Weyn, and Martin Klepal. 2009. Mobile phone-based displacement estimation for opportunistic localisation systems. In *Proceedings of the IEEE UBICOMM*.

[3] CamMeasure. 2017. Retrieved from https://itunes.apple.com/us/app.

[4] Si Chen, Muyuan Li, Kui Ren, Xinwen Fu, and Chunming Qiao. 2015. Rise of the indoor crowd: Reconstruction of building interior view via mobile crowdsourcing. In *Proceedings of the ACM SenSys*.
[5] Android developer. 2017. Retrieved from https://developer.android.com/index.html.
[6] Ruipeng Gao, Mingmin Zhao, Tao Ye, Fan Ye, Yizhou Wang, Kaigui Bian, Tao Wang, and Xiaoming Li. 2014. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *Proceedings of the ACM MobiCom*.
[7] Jinsong Han, Chen Qian, Panlong Yang, Dan Ma, Zhiping Jiang, Wei Xi, and Jizhong Zhao. 2016. GenePrint: Generic and accurate physical-layer identification for UHF RFID tags. *IEEE/ACM Trans. Netw.* 24, 2 (2016), 846–858.
[8] James Hays and Alexei A. Efros. 2008. IM2GPS: Estimating geographic information from a single image. In *Proceedings of the IEEE CVPR*.
[9] Jasper Jahn, Ulrich Batzer, Jochen Seitz, Lucila Patino-Studencka, and Javier Gutiérrez Boronat. 2010. Comparison and evaluation of acceleration-based step length estimators for handheld devices. In *Proceedings of the IEEE IPSN*.
[10] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2015. Overlay: Practical mobile augmented reality. In *Proceedings of the ACM MobiSys*.
[11] Xiaoying Kong. 2004. INS algorithm using quaternion model for low cost IMU. *Elsevier Robot. Auton. Syst.* 46, 4 (2004), 221–246.
[12] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. 2015. Spotfi: Decimeter level localization using wifi. In *Proceedings of the ACM SIGCOMM*.
[13] Ye-Sheng Kuo, Pat Pannuto, Ko-Jen Hsiao, and Prabal Dutta. 2014. Luxapose: Indoor positioning with mobile phones and visible light. In *Proceedings of the ACM MobiCom*.
[14] Liqun Li, Pan Hu, Chunyi Peng, Guobin Shen, and Feng Zhao. 2014. Epsilon: A visible light-based positioning system. In *Proceedings of the USENIX NSDI*.
[15] Tianxing Li, Chuankai An, Zhao Tian, Andrew T. Campbell, and Xia Zhou. 2015. Human sensing using visible light communication. In *Proceedings of the ACM MobiCom*.
[16] Sebastian O. H. Madgwick, Andrew J. L. Harrison, and Ravi Vaidyanathan. 2011. Estimation of IMU and MARG orientation using a gradient descent algorithm. In *Proceedings of the IEEE ICORR*.
[17] Justin Gregory Manweiler, Puneet Jain, and Romit Roy Choudhury. 2012. Satellites in our pockets: An object positioning system using smartphones. In *Proceedings of the ACM MobiSys*.
[18] João Luís Marins, Xiaoping Yun, Eric R. Bachmann, Robert B. McGhee, and Michael J. Zyda. 2001. An extended Kalman filter for quaternion-based orientation estimation using MARG sensors. In *Proceedings of the IEEE/RSJ IROS*.
[19] Smart Measure. 2017. Retrieved from https://play.google.com/store/apps.
[20] Shahriar Nirjon, Jie Liu, Gerald DeJean, Bodhi Priyantha, Yuzhe Jin, and Ted Hart. 2014. COIN-GPS: Indoor localization from direct GPS receiving. In *Proceedings of the ACM MobiSys*.
[21] Robin Wentao Ouyang, Animesh Srivastava, Prithvi Prabahar, Romit Roy Choudhury, Merideth Addicott, and F. Joseph McClernon. 2013. If you see something, swipe towards it: Crowdsourced event localization using smartphones. In *Proceedings of the ACM UbiComp*.
[22] Abhinav Parate, Meng-Chieh Chiu, Chaniel Chadowitz, Deepak Ganesan, and Evangelos Kalogerakis. 2014. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In *Proceedings of ACM MobiSys*.
[23] Nirupam Roy, He Wang, and Romit Roy Choudhury. 2014. I am a smartphone and i can tell my user's walking direction. In *Proceedings of the ACM MobiSys*.
[24] Kartik Sankaran, Minhui Zhu, Xiang Fa Guo, Akkihebbal L. Ananda, Mun Choon Chan, and Li-Shiuan Peh. 2014. Using mobile phone barometer for low-power transportation context detection. In *Proceedings of the ACM SenSys*.
[25] Longfei Shangguan, Zheng Yang, Alex X. Liu, Zimu Zhou, and Yunhao Liu. 2017. STPP: Spatial-temporal phase profiling-based method for relative RFID tag localization. *IEEE/ACM Trans. Netw.* 25, 1 (2017), 596–609.
[26] Gabriel Takacs, Vijay Chandrasekhar, Natasha Gelfand, Yingen Xiong, Wei-Chao Chen, Thanos Bismpigiannis, Radek Grzeszczuk, Kari Pulli, and Bernd Girod. 2008. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceedings of the ACM MIR*.
[27] Deepak Vasisht, Swarun Kumar, and Dina Katabi. 2016. Decimeter-level localization with a single wifi access point. In *Proceedings of the USENIX NSDI*.
[28] Daniel Wagner and Dieter Schmalstieg. 2009. Making augmented reality practical on mobile phones, part 1. *IEEE Comput. Graph. Appl.* 29, 3 (2009), 12–15.
[29] Ge Wang, Chen Qian, Longfei Shangguan, Han Ding, Jinsong Han, Nan Yang, Wei Xi, and Jizhong Zhao. 2017. HMRL: Relative localization of RFID tags with static devices. In *Proceedings of the IEEE SECON*.
[30] He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. Mole: Motion leaks through smartwatch sensors. In *Proceedings of the ACM MobiCom*.
[31] Pengjin Xie, Jingchao Feng, Zhichao Cao, and Jiliang Wang. 2017. GeneWave: Fast authentication and key agreement on commodity mobile devices. In *Proceedings of the IEEE ICNP*.

[32] Jie Xiong and Kyle Jamieson. 2013. ArrayTrack: A fine-grained indoor location system. In *Proceedings of the USENIX NSDI*.

[33] Zheng Yang, Zimu Zhou, and Yunhao Liu. 2013. From RSSI to CSI: Indoor localization via channel response. *Comput. Surveys* 46, 2 (2013), 25.

[34] Lan Zhang, Kebin Liu, Yonghang Jiang, Xiang-Yang Li, Yunhao Liu, Panlong Yang, and Zhenhua Li. 2017. Montage: Combine frames with movement continuity for realtime multi-user tracking. *IEEE Trans. Mobile Comput.* 16, 4 (2017), 1019–1031.

[35] Xiaolong Zheng, Jiliang Wang, Longfei Shangguan, Zimu Zhou, and Yunhao Liu. 2017. Design and implementation of a CSI-based ubiquitous smoking detection system. *IEEE/ACM Trans. Netw.* 25, 6 (2017), 3781–3793.

[36] Pengfei Zhou, Mo Li, and Guobin Shen. 2014. Use it free: Instantly knowing your phone attitude. In *Proceedings of the ACM MobiCom*.

Received June 2018; revised October 2018; accepted October 2018