IET Communications

Research Article

IET Journals
The Institution of Engineering and Technology

Pedestrian walking safety system based on smartphone built-in sensors

ISSN 1751-8628
Received on 22nd May 2017
Revised 7th December 2017
Accepted on 12th January 2018
doi: 10.1049/iet-com.2017.0502
www.ietdl.org

Yantao Li[1,2,3], Fengtao Xue[1], Xinqi Fan[1], Zehui Qu[1] ✉, Gang Zhou[2]

[1]*College of Computer and Information Sciences, Southwest University, Chongqing 400715, People's Republic of China*
[2]*Department of Computer Science, College of William and Mary, Williamsburg, VA 23185, USA*
[3]*State Key Laboratory for Novel Software Technology, Nanjing University, Jiangsu 210023, People's Republic of China*
✉ *E-mail: quzehui@swu.edu.cn*

Abstract: People watching smartphones while walking causes a significant impact to their safety. Pedestrians staring at smartphone screens while walking along the sidewalk are generally more at risk than other pedestrians not engaged in smartphone usage. In this study, the authors propose *Safe Walking*, an Android smartphone-based system that detects the walking behaviour of pedestrians by leveraging the sensors and front camera on smartphones, improving the safety of pedestrians staring at smartphone screens. More specifically, *Safe Walking* first exploits a pedestrian speed calculation algorithm by sampling acceleration data via the accelerometer and calculating gravity components via the gravity sensor. Then, this system utilises a greyscale image detection algorithm to detect the face and eye movement modes based on OpenCV4Android to determine if pedestrians are staring at the screens. Finally, *Safe Walking* generates a vibration by a vibrator on smartphones to alert pedestrians to pay attention to road conditions. The authors implemented *Safe Walking* on an Android smartphone and evaluated pedestrian walking speed, the accuracy of eye movement, and system performance. The results show that *Safe Walking* can prevent the potential danger for pedestrians staring at smartphone screens with a true positive rate of 91%.

1 Introduction

With the rapid development of communication technology, a variety of smartphones have been manufactured, and with the wide spread of smartphones, numerous smartphone-based applications have been developed, such as Facebook, YouTube, Twitter, Gmail and Instagram [1, 2]. Therefore, multimedia functionality gradually dominates the usage of smartphones. People prefer to use smartphones when they are walking because they are busy with working when staying in the office. Although walking provides physical and mental benefits, people watching smartphones while walking leads to injuries or traffic fatalities. As reported, pedestrian fatalities comprised 11% of all traffic deaths nationwide in 2005, but 15% in 2014 in the United States [3–5]. In most of the cases, both drivers and pedestrians are responsible for accidents. Moreover, pedestrians' use of smartphones for entertainment may lead to inattentional blindness, which is more likely to cause serious accidents [6, 7]. Therefore, the research question of this work is how to ensure the walking safety of pedestrian mobile phone users without any surrounding environment support.

To increase the safety of pedestrian mobile phone users, a variety of research works have been conducted, which can be categorised into pedestrian-surrounding safety studies and pedestrian-behaviour safety investigations. For pedestrian-surrounding safety studies, the authors propose *Surround-see* [8], *UltraSee* [9], *LookUp* [10], *FOSF* [11], *pSafety* [12], *CrowdWatch* [13], *Bumpalert* [14]. However, most of these systems mainly rely on surrounding environments of pedestrians, but do not address pedestrians' carelessness. For pedestrian-behaviour safety investigations, the authors present a face detection system [15] and a gait detection system [16]. However, these approaches are dedicated to specific scenarios.

While existing approaches address various specific aspects, their reliance on surrounding environments of pedestrians, such as vehicle distance and ground change, prevent them from detecting the walking behaviour of pedestrians. To fill this gap, we propose *Safe Walking*, an Android smartphone-based system that detects the walking behaviour of pedestrians by leveraging smartphone sensors and the front camera, improving the safety of pedestrians staring at smartphone screens. *Safe Walking* estimates the walking speed of pedestrians and uses the front camera of the mobile phone to capture the staring period of smartphone users, alerting them of a potentially unsafe situation. More specifically, *Safe Walking* estimates the moving speed of pedestrians by using the smartphone's gravity sensor and accelerometer sensor. When the walking speed reaches a threshold of normal walking speed, *Safe Walking* activates the front camera based on OpenCV4Android to capture the pedestrian staring duration at the smartphone screen. If the staring period reaches the threshold of normal watching time, *Safe Walking* alerts pedestrians via a vibration to aid walking. We implemented *Safe Walking* on an Android smartphone and evaluated pedestrian walking speed, accuracy of eye movement, and system performance. The results show that it can prevent the potential danger for pedestrians staring at smartphones.

The main contributions of this work can be summarised as follows:

- We design *Safe Walking*, an Android smartphone-based system that detects the walking behaviours of pedestrians by leveraging smartphone sensors and the front camera, improving the safety of pedestrians staring at smartphone screens.
- We propose a pedestrian speed calculation algorithm by sampling acceleration data via the accelerometer and calculating gravity components via the gravity sensor.
- We utilise a greyscale image detection algorithm to detect face and eye movement modes based on OpenCV4Android to determine if the pedestrian is staring at the screen.
- We implement *Safe Walking* on an Android smartphone and evaluate pedestrian walking speed, the accuracy of eye movement, and system performance.

The remainder of this work is organised as follows: Section 2 provides the related research on driver-side driving behaviour detection and pedestrian-side walking behaviour detection. Section 3 describes the system architecture of *Safe Walking* in modules of walking speed estimation, face and eye detection, and pedestrian alerts. Performance of *Safe Walking* is evaluated in Section 4, and

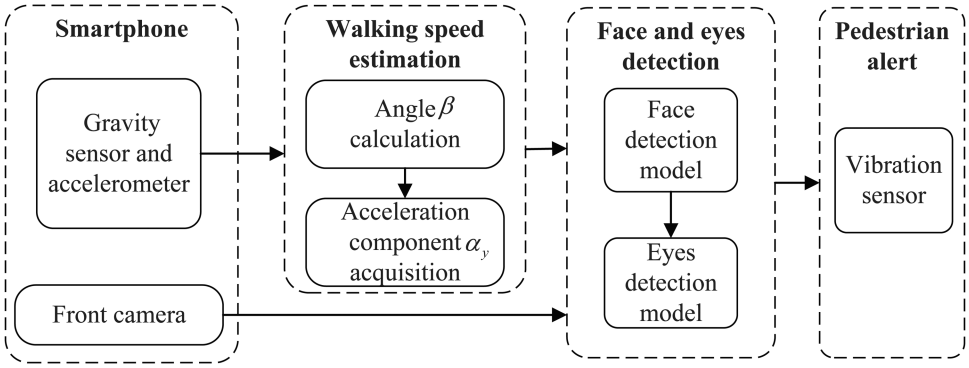


Fig. 1 *Safe Walking system architecture*

we summarise the achievements of this work and discuss the future work in Section 5.

2 Related work

There is a wide range of research studies focusing on walking safety of pedestrian smartphone users, which can be categorised into the pedestrian-surrounding safety studies and the pedestrian-behaviour safety investigations.

Studies of pedestrian-surrounding safety have been conducted, which sense surrounding environments of pedestrians and provide active feedbacks. The authors of [8] present *Surround-See*, a self-contained smartphone equipped with an omni-directional camera that enables peripheral vision around the device to augment daily mobile tasks. In [9], the authors propose *UltraSee* to detect the sudden change of ground for pedestrian mobile phone users, augmenting smartphones with a small ultrasonic sensor which can detect the abrupt change of distance ahead. The authors of [10] design *LookUp* that uses shoe-mounted inertial sensors for location classification based on the surface gradient profile and step patterns. In [11], the authors propose a pedestrian-oriented forewarning system *FOSF* to provide forewarning alert for smartphone users using the data including location, speed, and travel direction collected from smartphones. The authors of [12] implement *pSafety*, a collision prevention system instantaneously informs pedestrians and drivers of the potential threatening accidents. This system collects global positioning system (GPS) information from smartphones of pedestrians and vehicle drivers through mobile networks. In [13], the authors propose *CrowdWatch* which leverages mobile crowd sensing and crowd intelligence aggregation to detect temporary obstacles and make effective alerts for distracted pedestrians. The authors of [14] present *Bumpalert* that provides a generic solution without requiring any prior knowledge of the user's environment by estimating distances to nearby objects using the phones' speakers and microphones. However, most of these systems mainly rely on surrounding environments of pedestrians including the vehicle or object distance, ground change, and obstacles on the road, but do not address pedestrians' carelessness.

A few investigations of pedestrian-behaviour safety have been conducted, which detect the behaviours of pedestrians and provide active alerts. The authors of [15] propose a distinctive feature, joint Haar-like feature, for detecting faces, and this feature can be calculated very fast and has robustness against the addition of noise and change in illumination. In [16], the authors introduce the architecture of a hierarchical face and eye detection system using the Haar cascade classifiers augmented with some simple knowledge-based rules. However, these systems are dedicated to specific scenarios for pedestrians, such as face detection and eye detection.

Without any surrounding environment support, we propose an Android smartphone-based system, *Safe Walking*, which detects the pedestrian's watching behaviour when walking using the energy-efficient accelerometer and front camera on smartphones.

3 System design

In this section, we describe the system architecture of *Safe Walking* in detail, as illustrated in Fig. 1. As shown in Fig. 1, *Safe Walking* is composed of three modules: (i) walking speed estimation; (ii) face and eye detection; and (iii) pedestrian alerts.

Safe Walking solely resides on an Android smartphone platform without any reliance on a backend server. When pedestrians begin walking, *Safe Walking* starts to estimate the walking speed of pedestrians by leveraging the accelerometer and the gravity sensor, then captures images via the front camera to detect pedestrian's face and eye movement modes based on OpenCV4Android. It alerts the pedestrians when they stare at the screen for a fixed time over a specific walking speed. More specifically, in the walking speed estimation module, the main task is to determine whether a pedestrian is walking and what the speed is. We propose a pedestrian speed calculation algorithm by sampling acceleration data via the accelerometer and calculating gravity components via the gravity sensor. In face and eye detection, we utilise a greyscale image detection algorithm to detect face and eye movement mode based on OpenCV4Android to determine if the pedestrian is staring at the screen. In the pedestrian alert module, a vibration will be generated to alert the participant to pay attention to road conditions. In the following, we will first detail the walking speed estimation module, and face and eye detection module of the system architecture of *Safe Walking*, and then describe the flowchart.

3.1 Walking speed estimation

In this section, the walking speed estimation module in *Safe Walking* will estimate pedestrian walking speed by leveraging the accelerometer and the gravity sensor on smartphones. First, we introduce the smartphone coordinate system, gravity decomposition, and sensor coordinate system, respectively. As illustrated in Fig. 2*a*, the smartphone coordinate system can be defined as: the *x*-axis of the phone is horizontal and points to the right side, the *y*-axis is vertical and points upward, and the *z*-axis is perpendicular to the *xy*-plane (composed of *x*-axis and *y*-axis) and points from the back to the front of smartphone's screen. Pedestrians chronically hold smartphones with screens towards their faces when reading news, watching videos or checking emails, and smartphones thus naturally form an angle with horizontal level or vertical level, where we define the angle β between the phone's *y*-axis and the walking direction, as demonstrated in Fig. 2*b*. The gravity of a phone always exists and points downward, no matter how it is held and whether the holder is in walking or stationary state. According to the smartphone coordinate system, the gravity *g* can be decomposed into g_y and g_z, as illustrated in Fig. 2*c*. In the Android operating system, the sensor coordinate system is defined as the *x*-axis is horizontal and points to the right, the *y*-axis is vertical and points up, and the *z*-axis points towards the outside of the screen face [17], which is the same as the smartphone coordinate system. For the acceleration sensor coordinate system, the corresponding accelerations of three-axis components are a_x, a_y and a_z.

How does *Safe Walking* estimate pedestrian walking speed? When pedestrians start walking, the walking speed estimation module will first register the accelerometer by the function

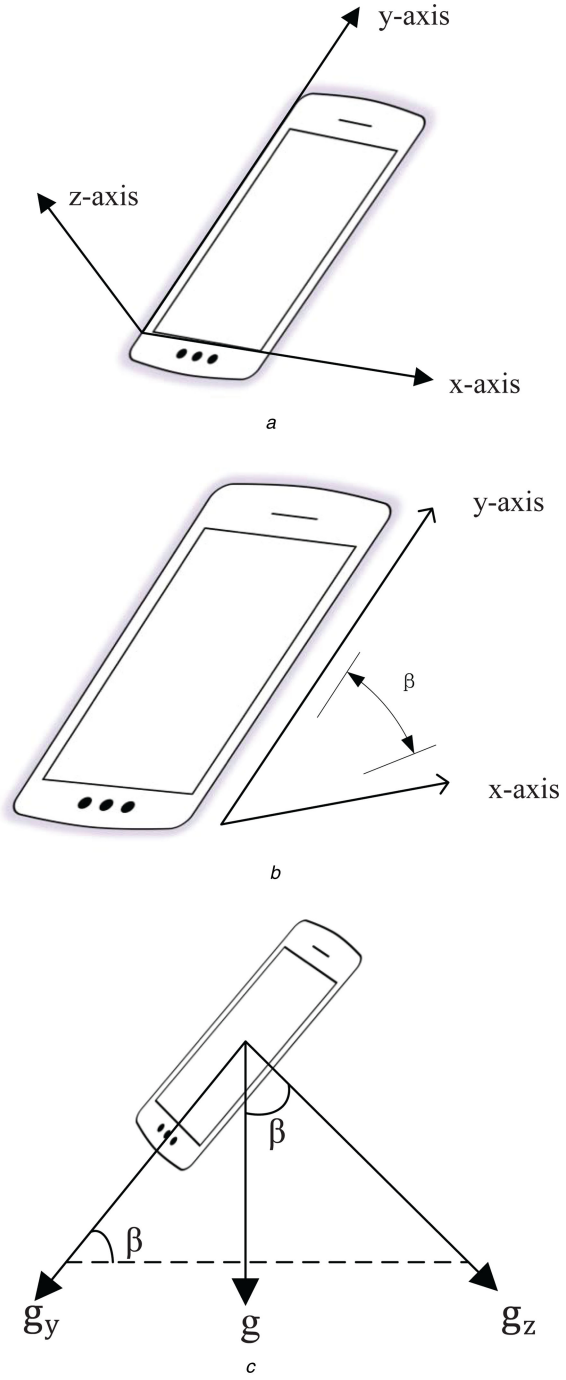


Fig. 2 *Smartphone coordinate system and gravity decomposition*
(a) Smartphone coordinate system, ***(b)*** Definition of angle β, ***(c)*** Gravity decomposition based on β

registerListener() of *SensorManager*, then listens to the sensor change by the function *onSensorChanged*(), and finally obtains three components of acceleration of the smartphone: a_x, a_y and a_z. Then, the angle β can be calculated by the two components of gravity g_y and g_z as shown in the following equations:

$$\tan\beta = \frac{g_y}{g_z}, \tag{1}$$

$$\beta = \tan^{-1}\left(\frac{g_y}{g_z}\right). \tag{2}$$

Next, walking acceleration a_{walking} can be calculated by using the following equation:

$$a_{\text{walking}} = a_y \times \cos\beta\,. \tag{3}$$

Finally, walking speed v_{walking} is calculated by integration of a_{walking} with the corresponding time interval as expressed in the following equation:

$$v_{\text{walking}}(t_1) = v_{\text{walking}}(t_0) + \int_{t_0}^{t_1} a_{\text{walking}}(t)\,\mathrm{d}t, \tag{4}$$

where $v_{\text{walking}}(t_1)$ is the pedestrian walking speed at time t_1, and $v_{\text{walking}}(t_0)$ is the pedestrian walking speed at time t_0. If $t_0 = 0$, that is, the pedestrian starts walking at time t_0, then $v_{\text{walking}}(t_0) = 0$. $a_{\text{walking}}(t)$ is the acceleration data sampled from time t_0 and t_1.

In implementation, since the acceleration sampling is a discrete value at a specific frequency, rather than a continuous value, we modify the pedestrian walking speed from (4) to (5)

$$v_{\text{walking}}(t_1) = v_{\text{walking}}(t_0) + \sum_{i=0}^{(t_1 - t_0)k} \frac{a_{\text{walking}}(i)}{k}, \tag{5}$$

where k indicates the sampling frequency of the smartphone accelerometer, and $a_{\text{walking}}(i)$ is the ith sampling value, that is, the ith reception of the acceleration value in the y-axis.

We describe pedestrian walking speed calculation in Algorithm 1 (see Fig. 3).

3.2 Face and eye detection

When the walking speed of a pedestrian exceeds the threshold of a specific speed (such as 1.2 mph) for a fixed period (such as 6 s), *Safe Walking* starts the face and eye detection module, which comprises four phases: (i) front-camera image capture; (ii) face orientation; (iii) eye focus; (iv) eye movement analysis. In this section, the face and eye detection module of *Safe Walking* will detect the face orientation and the eye focus, which is developed by using the Adaboost algorithm [18–22] and OpenCV4Android library [23]. Then, we use the greyscale image detection algorithm to determine the accurate eye movement mode of the detected pedestrian.

3.2.1 Front-camera image capture.: During the image capture phase, *Safe Walking* continuously captures images by using the front camera on a smartphone and then saves them on the smartphone storage for the following phases. The front-camera sensor invokes an intent to capture images with a fixed frequency using the *startActivityForResult*() function, and then exploits the *getExternalStoragePublicDirectory*() function to save images on the smartphone storage.

3.2.2 Face orientation.: *Safe Walking* processes these captured images to detect the face orientation of the pedestrian by (i) loading the cascade classifier *lbpcacade_frontalface.xml* from a local resource file, (ii) using the front camera to capture the canvas and detect the pedestrian's face, and (iii) releasing the cascade classifier. The corresponding results of face detection are illustrated in Fig. 4*a*, which indicates the pedestrian's face is oriented to the phone screen.

3.2.3 Eye focus.: Given the pedestrian's face is oriented to a smartphone screen, *Safe Walking* next captures the pedestrian's eye focus. According to facial features, the width of an eye of a normal person is about one-fifth of the width of the face, and the height of an eye is about one-seventh of that of the face. Assuming that w denotes the width of a normal face, and h indicates the height of the face, we first use two proper rectangles to label the left and right eyes, to roughly locate the eye focus. As demonstrated in Figs. 4*b–d*, the width of the rectangle, therefore, is $w/4$, and the height of that is $h/7$. Then, we improve the accuracy of the eye movement mode inside the two rectangles by (i) loading the left and right cascade classifiers *haarcascade_eye_left.xml*, *haarcascade_eye_right.xml* from the local resource file, (ii) detecting the eye in the rectangle area, and (iii) releasing the

cascade classifiers. The corresponding results of eye focus in Figs. 4*b–d* indicate the eyes turning right, staring at the screen, and turning left, respectively. We detect the corner points of the eye image by using the Harris corner detection algorithm [24–26].

In general, the corner on the edge of the image has the largest curvature change rate, and the majority of corners are on the edge. The corner detection method based on the difference of local greyscale of images is mainly used in pixel points analysis, and the classical algorithms are Harris corner detection and Susan corner detection [27]. Harris corner detection is a famous corner detection method which is built on the image's greyscale gradient. Using this intuitive physical phenomenon, Harris eye point detection exploits greyscale information differential in the pixel area to detect the difference between two images, and further determines the corner point, through moving the window in all directions on the image.

The corner detection corresponding to R, can be calculated by using the following equations:

$$R = \det(M) - k \times (\mathrm{tr}(M)^2), \tag{6}$$

$$\det(M) = \lambda_1 \times \lambda_2, \tag{7}$$

$$\mathrm{tr}(M) = \lambda_1 + \lambda_2, \tag{8}$$

where λ_1 and λ_2 are eigenvalues of the matrix $\boldsymbol{M}$, and $k = 0.05$. The matrix $\boldsymbol{M}$ is defined by

$$\boldsymbol{M} = \begin{bmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{bmatrix},$$

Input: Pedestrian starts walking at time t_0
Output: Pedestrian walking speed $v_{walking}(t_1)$

Set $v_{walking}(t_0) = 0$;
Acquire acceleration components a_x, a_y and a_z by $onSensorChanged()$ at time t_0;
Acquire gravity components g_y and g_z by gravity decomposition;
Compute $\beta = tan^{-1}(\frac{g_y}{g_z})$;
if a_x, a_y, *and* a_z *increase* **then**
    Compute $a_{walking} = a_y \times cos\beta$;
    Compute $v_{walking}(t_1) = v_{walking}(t_0) + \sum_{i=0}^{(t_1-t_0)k} \frac{a_{walking}(i)}{k}$;
else
    return;
end
return $v_{walking}(t_1)$

Fig. 3 *Algorithm 1: Pedestrian walking speed calculation*

where I_x denotes the partial derivative of the direction x, I_y represents the partial derivative at the direction y, and I_{xy} is the mixed partial derivative.

The Harris corner detection algorithm calculates the R value of the corner point, then removes some points that have a small R value, and finally reserves points with bigger R as the potential corners. By using the Harris corner detection algorithm to detect the corner points of the eye image, the corresponding results are illustrated in Fig. 5*a*. From our practical experiments, we find that the number of corner points exceeds the number we expected. According to the relative positions we searched, the corner points are close to the edge of the triangle. Hence, we select seven points near the left and right edges of the eye image, respectively, to make the system more efficient, as demonstrated in Fig. 5*b*. Then, we calculate the mean value of the number of left and right corners as the optimal eye corner point.

After the optimal eye corner is obtained, the main task of eye focus is to recognise the eye movement mode, which is determined by the eyeball's location. We use the Canny edge detection algorithm to detect the eyes' edge [28, 29], modelling a closed region composed of the upper edge and left and right corners, as shown in Fig. 6*a*. In Fig. 6*a*, point A is the eye's left corner, while point D is the eye's right corner. Points B and C split line AD into three equivalent parts. We define the closed region as three regions a, b and c, and the corresponding eye movement modes are mode a (turn left), mode b (look forward), mode c (turn right). In a greyscale image, the greyscale value of pixels ranges in $[\,0, 255]$ [30], where black corresponds to low values while white associates with high values. We set a greyscale threshold $T(T = 60)$ to determine which part the eye is in, which corresponds to the eyes movement mode. In the greyscale picture, the eyeball has a lower grey value; therefore, if the eyeball is inside a specific part, the number of the pixel with greyscale will be less than the threshold T. More specifically, (i) if the grey value of the pixel is lower than the threshold T in the right closed region and is more than that of the middle and left regions, the eyeball is viewed as in right region a, corresponding to eye movement mode a. (ii) If the grey value of the pixel is lower than threshold T in the middle region and is more than that of the other two regions, the eyeball is viewed as in middle region b, corresponding to eye movement mode b. (iii) If the grey value of the pixel is lower than threshold T in the left region and is more than that of the other two regions, the eyeball is viewed as in the left region b, corresponding to eye movement mode c. As a consequence, our system aims to detect the mode b, because it indicates that the pedestrian is looking forward, that is, the pedestrian is staring at the smartphone's screen. As a result, Fig. 6*b* shows eye movement mode a, mode b and mode c, respectively.

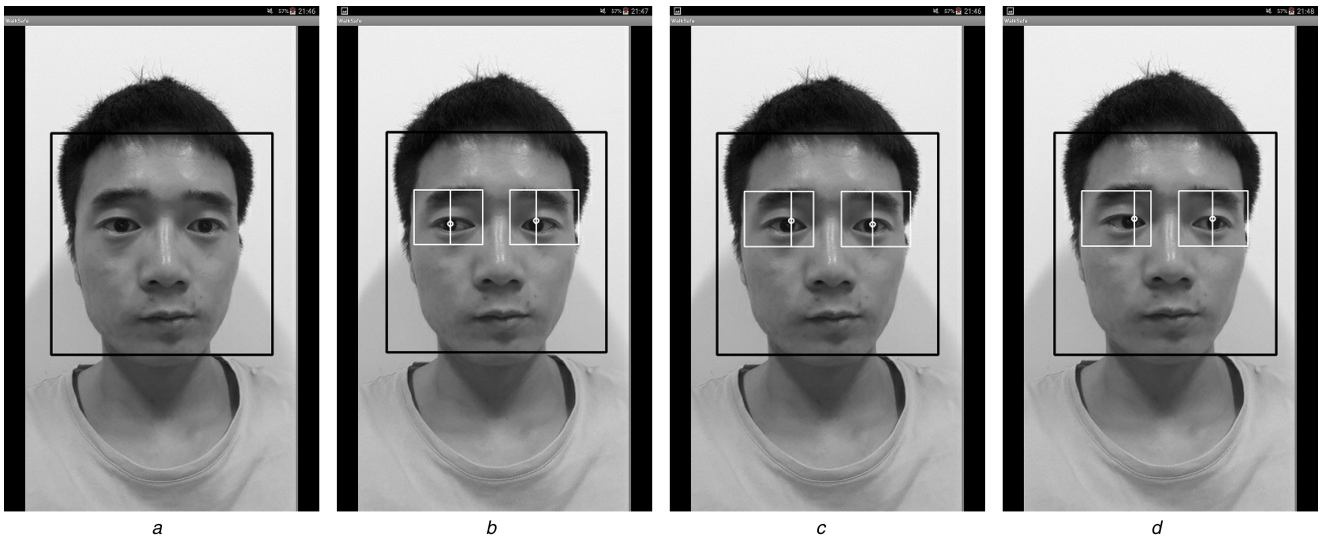
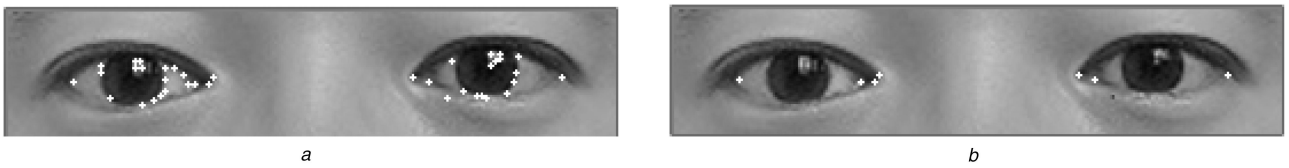
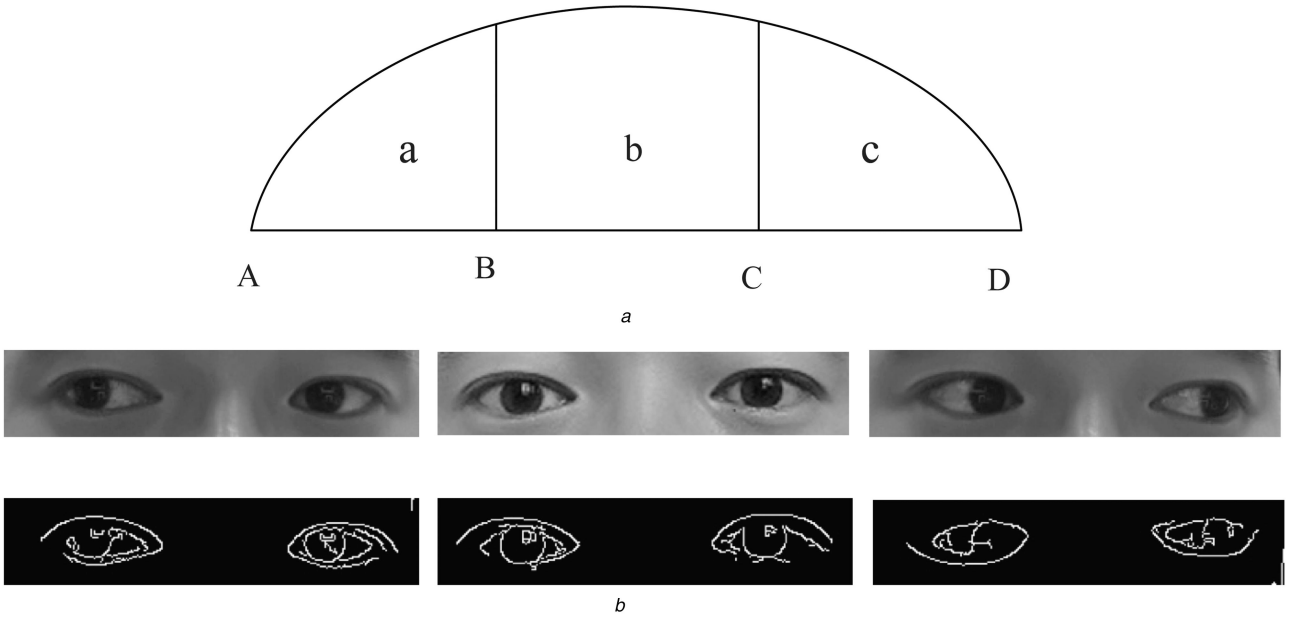


Fig. 4 *Face and eye detection*
(a) Face capture, ***(b)*** Eyes turning right, ***(c)*** Eyes on screen (looking forward), ***(d)*** Eyes turning left



a *b*

Fig. 5 *Eye focus based on Harris corner algorithm*
(a) Original results, ***(b)*** More efficient results



a *b*

Fig. 6 *Eye movements*
(a) Eye model, ***(b)*** Left column: mode *a*; middle column: mode *b*; right column: mode *c*

4 Performance evaluation

In this section, we evaluate the performance of *Safe Walking* on real implementation. We use OpenCV (Open Source Computer Vision Library) to implement *Safe Walking*, where OpenCV4Android is available as an SDK with a set of samples and Javadoc documentation for OpenCV Java API [23]. In the following, we first describe the evaluation setup, and then evaluate the performance in terms of pedestrian walking speed, the accuracy of eye movement, and the system, respectively.

4.1 Evaluation setup

In the evaluation, we developed *Safe Walking* on an Android platform, using SDK version 18 and version 4.3 of the operating system. We solely implemented *Safe Walking* on a Samsung Galaxy *S*4 smartphone, which was equipped with Android 4.2.2 (Jelly Bean) operating system, Exynos 5410 Octa-core (4×1.6 GHz Cortex-A15 and 4×1.2 GHz Cortex-A7) CPU and 2 GB RAM memory, an accelerometer with a maximum sampling rate of 200 GHz and a front-facing camera with 13 MP resolution. We leverage the OpenCV4Android library for computer vision algorithms.

4.2 Analysis of pedestrian walking speed

To evaluate pedestrian walking speed, we conducted real tests with the implementation of *Safe Walking* on the smartphone, where a participant was watching news on the smartphone screen while walking. We plotted the participant's acceleration with a sampling rate of 20 Hz in Fig. 7*a*, where the *Y*-axis and *Z*-axis are the accelerations in the smartphone coordinate system. As illustrated in Fig. 7*a*, before walking, the accelerations of the *Y*-axis and *Z*-axis remain in a steady state; they fluctuate with the participant walking, and steady state recurs when they stop walking. Moreover, the amplitude of the *Y*-axis is around 0.6 and that of the *Z*-axis is about 10. The *Y*-axis shows larger amplitude than the *Z*-axis because there is always gravity in the *Z*-axis of the smartphone. When the participant starts walking, stepping up and down impact *Z*-axis.

Furthermore, we investigate pedestrian walking steps to figure out the walking speed, based on the accelerations of pedestrian walking fluctuation. As shown in Fig. 7*b*, when the participant steps up, *Y*-axis acceleration gradually increases and it decreases gradually when the participant steps down, which is a step cycle period, ~ 0.5 s. In the implementation, we choose the first half of a step cycle period as the time for calculating an original acceleration because the second half period is usually too short and the corresponding acceleration decreases dramatically, which is not suitable for data collection. Based on the collected acceleration data, we fit a pedestrian acceleration curve, as illustrated in Fig. 7*c*. According to curve fitting, we can obtain the acceleration for each step. Based on the acceleration and (5), *Safe Walking* can calculate the pedestrian walking speed, as indicated in Fig. 7*d*.

4.3 Accuracy of eye movement

We evaluated the accuracy of eye movement when the participant was holding a smartphone with about 30 cm from eyes to the screen. The participant watched right, forward, and left for 6 s respectively, and *Safe Walking* captured the participant's face and eyes' information, as illustrated in Figs. 8*a*–*c*, respectively. We analysed the eye movement and plotted the distribution of grey values in areas *a*, *b*, and *c* in Fig. 8*d*. As illustrated in the left of Fig. 8*d*, the number of pixels of greyscale lower than the threshold T ($T = 60$) in the left region is more than that in the middle and right regions, which indicates that the participant turns his eyes to the left. The number of pixels of greyscale lower than threshold T in the middle region is more than that of the other two regions, which indicates that the participant is watching on the screen, as demonstrated in the middle of Fig. 8*d*. The number of pixels of the greyscale lower than threshold T in the right region is more than that of the other two regions, which indicates the participant turns his eyes right, as shown in the right of Fig. 8*d*.

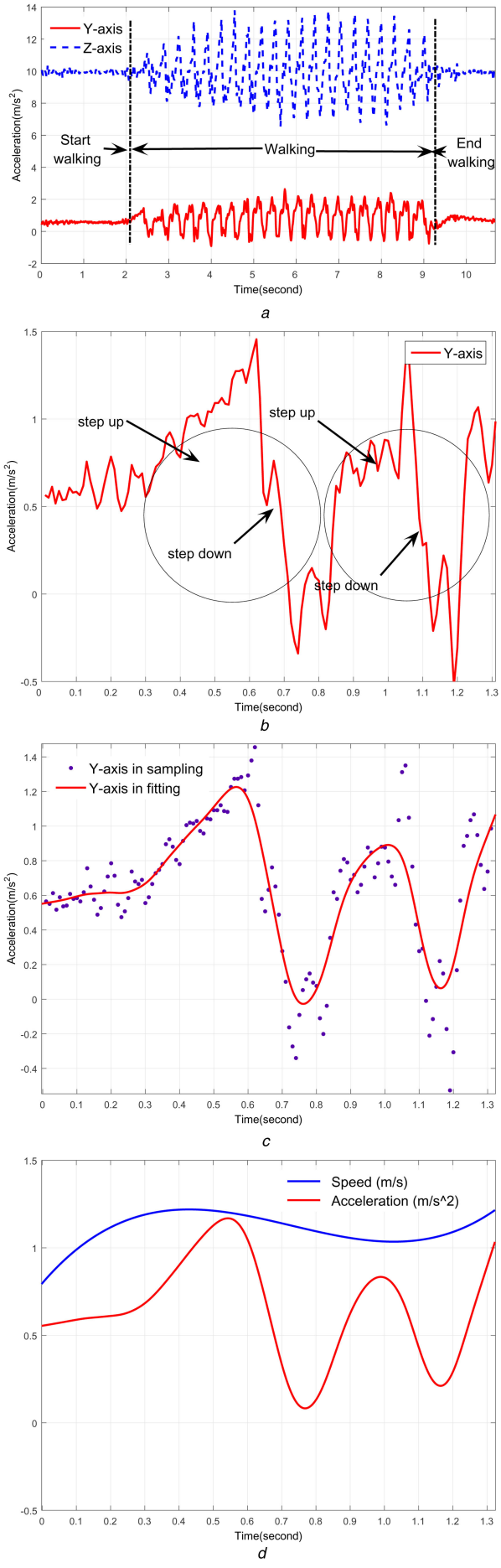


Fig. 7 *Walking acceleration analysis*
(a) Walking acceleration in the *YZ*-axis, ***(b)*** Walking acceleration in the *Y* axis, ***(c)*** Acceleration fitting, ***(d)*** Acceleration and speed

Therefore, the evaluation results (Fig. 8*d*) are completely consistent with the participant's activity (Figs. 8*a–c*), which demonstrates that the eye movement detection in *Safe Walking* is highly accurate.

4.4 System evaluation

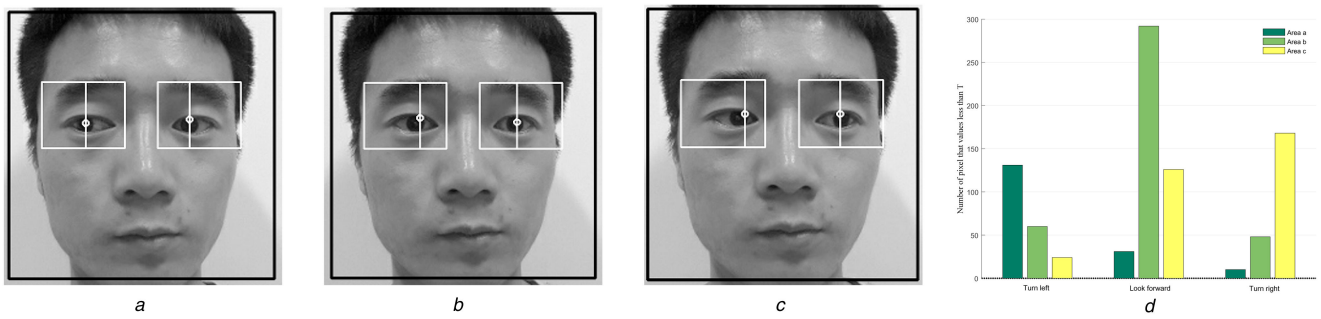
To determine the thresholds of the walking speed and staring period of pedestrians, we randomly recorded pedestrians staring at smartphones while walking on sidewalks, and also conducted a survey of young people who happened to watch smartphones while walking on the road. Based on the observations, we found: (i) the walking speed was normally around 1.2 mph, and (ii) if people stared at their smartphone screens for over 6 s, they deviated from the original direction and hesitated to walk forward or suspended to adjust the view of sight. Therefore, we set the threshold of the walking speed to be 1.2 mph and the threshold of the staring period to be 6 s.

To evaluate the performance of *Safe Walking*, we conducted a random user study by 16 participants staring at screens while walking with the implementation of *Safe Walking* on smartphones. *Safe Walking* recorded the participants' eye images when they were staring at the screens, and calculated the grey value of the pupils as illustrated in Fig. 9. In Fig. 9, the *x* scale indicates the 16 participants, and the *y* scale represents the pixel number of left and right corners of eyeballs. As we can see, Fig. 9*a* shows that the number of corner points in region *b* is more than that in regions *a* and *c*, which means the participants are staring at the screens. If the time period of a desired number of greyscale in region *b* exceeds 6 s, *Safe Walking* will send an alert to the participant. In contrast, Figs. 9*b* and *c* show the participants are looking at surroundings, rather than the screens, so the participants are safe in this case. Moreover, based on the general behaviour of smartphone usage, the distance between users' eyes and their smartphone screens is typically about 30 cm, which indicates that *Safe Walking* can accurately detect the face and eye movement through the front camera with this proximity.

To evaluate the accuracy of *Safe Walking*, we carried out a number of real world experiments: a 32-min pedestrian smartphone user experiment was performed by the 16 participants who were holding smartphones while walking on the sidewalks in the campus, where each participant took 2 min. We recorded the video captured by a Samsung Galaxy *S*4 smartphone and counted the number of alerts that should be received by participants in the video as a ground truth. We compared the ground truth with the number of alerts sent by *Safe Walking*. In this evaluation, we define a positive alert as the participant stays in the state of staring at screens while walking more than 6 s. Any alert when the participant stays in that state more than 6 s is classified as a successful alert (true positives) and the ratio of these alerts is called the true positive rate. On the other hand, any alert occurring when the participant is actually safe is classified as a false alert (false positives), and the corresponding rate is calculated as the false positive rate. Based on the ground truth, in the 32-min experiment, the participants should have been alerted 296 times. In fact, they were alerted 281 times. Of the 281 alerts, 268 alerts were identified (true positives) and 13 were false (false positives). The corresponding results are shown in Table 1. As illustrated in Table 1, *Safe Walking* shows a true positive rate of 91%, which is sufficient to alert most of the risks that pedestrians meet, and a false positive rate of 5%.

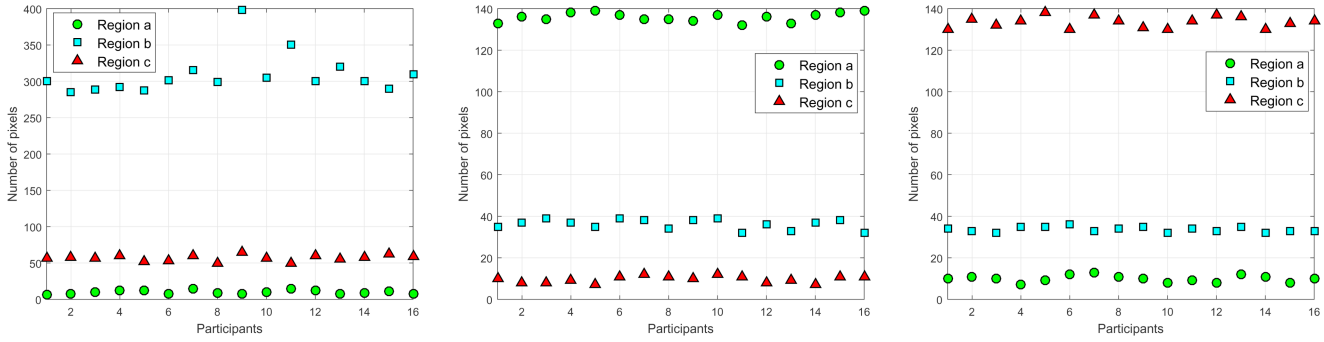
4.5 Comparison with related systems

To compare *Safe Walking* with related systems, we list five representative pedestrian safety systems in Table 2 with the system name in the first column, the function provided by these systems in the middle, and the sensor utilised in the last. As shown in Table 2, *Safe Walking* provides detection of walking safety while watching smartphone screens for pedestrians with the energy-efficient accelerometer and front camera on smartphones without any surrounding environment support. Representative systems either need peripheral environment and objects, ground conditions, or



a *b* *c* *d*

Fig. 8 *Eye's movement*
(a) Turn left, ***(b)*** Look forward, ***(c)*** Turn right, ***(d)*** Number of pixels in eyes' area



a *b* *c*

Fig. 9 *Pixel number of 16 participants*
(a) Staring at the screen (mode *b*), ***(b)*** Turning left (mode *a*), ***(c)*** Turning right (mode *c*)

Table 1 Accuracy of *Safe Walking* in a 32-min experiment

Alert	Value
number of alerts in ground truth	296
number of alerts occurred	281
number of successful alerts	268
number of false alerts	13
true positive rate	91%
false positive rate	5%

Table 2 Related systems comparison

System	Function	Sensor
Surround-see [8]	a peripheral vision system for recognising device's peripheral environment and objects	omni-directional camera, front-facing camera
UltraSee [9]	an unobtrusive alertness system for detecting sudden change of ground	ultrasonic sensor
LookUp [10]	a pedestrian safety service system focusing on dead reckoning and inertial navigation	shoe-mounted inertial sensors
pSafety [12]	a collision prevention system for pedestrians using smartphones	GPS
CrowdWatch [13]	a dynamic sidewalk obstacle detection system using mobile crowd sensing	accelerometer, orientation sensor, camera, GPS
Safe Walking	a smartphone-based system for detecting pedestrians' walking behaviours	accelerometer, front camera

obstacles on the road with power-hungry sensors, such as GPS and ultrasonic sensor.

5 Conclusions and future work

Safe Walking is an Android smartphone-based system that detects the walking behaviour of pedestrians by leveraging the sensors and front camera on smartphones, improving the safety of pedestrians staring on smartphone screens. We propose a pedestrian speed calculation algorithm by sampling acceleration data via the accelerometer and calculating gravity components via the gravity sensor. We utilise a greyscale image detection algorithm to detect face and eye movement modes based on OpenCV4Android to determine if pedestrians are staring at the screens. We implemented *Safe Walking* on an Android smartphone and then evaluated pedestrian walking speed, the accuracy of eye movement, and system performance.

In the future, we will design a more effective face and eye detection algorithm and consider the energy consumption of smartphones. We will also develop an efficient solution for blind people who walk on the street alone.

6 Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (grant nos. 61672119, 61528206, and 61402380), the Natural Science Foundation of CQ CSTC (grant no. cstc2015jcyjA40044), the U.S. National Science Foundation (grant nos. CNS-1253506(CAREER) and CNS-1618300), and the Opening Project of State Key Laboratory for Novel Software Technology (grant no. KFKT2016B13).

7 References

[1] Heisler, Y.: 'List of most popular iOS and Andriod apps in 2016'. Available at: http://bgr.com/2016/02/04/most-popular-smartphone-apps-facebook-google/, accessed 15 November 2016

[2] Zhang, Q., Yang, L.T., Chen, Z.: 'Deep computation model for unsupervised feature learning on big data', *IEEE Trans. Serv. Comput.*, 2016, **9**, (1), pp. 161–171

[3] Governors Highway Safety Association: 'Pedestrian traffic fatalities by state: 2015 preliminary data'. Available at: http://www.ghsa.org/files/pubs/spotlights/spotlight_ped2015.pdf, accessed 20 November 2016

[4] Lopresti-Goodman, S.M., Rivera, A., Dressel, C.: 'Practicing safe text: the impact of texting on walking behavior', *Appl. Cogn. Psychol.*, 2012, **26**, (4), pp. 644–648

[5] Zhang, Q., Zhong, H., Yang, L.T., *et al.*: 'PPHOCFS: privacy preserving high-order CFS algorithm on cloud for clustering multimedia data', *ACM Trans. Multimedia Comput. Commun. Appl.*, 2016, **12**, (4s), Article No. 66, pp. 1–15
[6] Gandhi, T., Trivedi, M.: 'Pedestrian protection systems: issues, survey, and challenges', *IEEE Trans. Intell. Transp. Syst.*, 2007, **8**, (3), pp. 413–430
[7] Hyman, I.E., Boss, S.M., Wise, B.M., *et al.*: 'Did you see the unicycling clown? Inattentional blindness while walking and talking on a cell phone', *Appl. Cogn. Psychol.*, 2010, **24**, (5), pp. 597–607
[8] Yang, X.-D., Hasan, K., Bruce, N., *et al.*: 'Surround-see: enabling peripheral vision on smartphones during active use'. Proc. 26th ACM Symp. on User Interface Software and Technology, St. Andrews, Scotland, UK, October 2013, pp. 291–300
[9] Wen, J., Cao, J., Liu, X.: 'We help you watch your steps: unobtrusive alertness system for pedestrian mobile phone users'. Proc. IEEE Int. Conf. on Pervasive Computing and Communications, St. Louis, MO, USA, March 2015, pp. 105–113
[10] Jain, S., Borgiattino, C., Ren, Y.: 'LookUp: enabling pedestrian safety services via shoe sensing'. Proc. 13th Int. Conf. on Mobile Systems, Applications, and Services, Florence, Italy, May 2015, pp. 257–271
[11] Liu, Z., Pu, L., Meng, Z.: 'POFS: a novel pedestrian-oriented forewarning system for vulnerable pedestrian safety'. Proc. Int. Conf. on Connected Vehicles and Expo, Shenzhen, China, October 2015, pp. 100–105
[12] Lin, C., Chen, Y., Chen, J., *et al.*: 'pSafety: a collision prevention system for pedestrians using smartphone'. Proc. IEEE 84th Vehicular Technology Conf., Montreal, QC, Canada, September 2016, pp. 1–5
[13] Wang, Q., Guo, B., Wang, L., *et al.*: 'CrowdWatch: dynamic sidewalk obstacle detection using mobile crowd sensing', *IEEE Internet Things J.*, 2017, **4**, (6), pp. 2159–2171
[14] Tung, Y., Shin, K.G.: 'Use of phone sensors to enhance distracted pedestrians' safety', *IEEE Trans. Mobile Comput.*, 2017, DOI: 10.1109/TMC.2017.2764909, pp. 1–14
[15] Mita, T., Kaneko, T., Hori, O.: 'Joint Haar-like features for face detection'. Proc. 10th IEEE Int. Conf. on Computer, Beijing, China, October 2005, pp. 1–8
[16] Kasinski, A., Schmidt, A.: 'The architecture and performance of the face and eyes detection system based on the Haar cascade classifiers', *Pattern Anal. Appl.*, 2010, **13**, (2), pp. 197–211
[17] Android Developers: Available at: http://www.android-doc.com/guide/topics/sensors/sensors_overview.html, accessed 09 February 2016
[18] Viola, P., Jones, M.J.: 'Robust real-time object detection', *Int. J. Comput. Vis.*, 2004, **57**, (2), pp. 137–154
[19] Zhang, Q., Yang, L.T., Chen, Z.: 'Privacy preserving deep computation model on cloud for big data feature learning', *IEEE Trans. Comput.*, 2016, **65**, (5), pp. 1351–1362
[20] Viola, P., Jones, M.: 'Rapid object detection using a boosted cascade of simple features'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Kauai, HI, USA, December 2001, pp. 511–518
[21] Feick, R., Ahumada, L., Carroasco, H.: 'Effect of pedestrian traffic on fade statistics of fixed wireless links in public spaces', *IET Commun.*, 2011, **5**, (16), pp. 2258–2290
[22] Wang, T., Cardone, G., Corradi, A., *et al.*: 'WalkSafe: a pedestrian safety app for mobile phone users who walk and talk while crossing roads'. Proc. 12th Workshop on Mobile Computing Systems and Applications, San Diego, CA, USA, February 2012, pp. 1–6
[23] OpenCV: Available at: http://opencv.org/platforms/android.html, accessed 22 April 2016
[24] Ryu, J.B., Park, H.H., Park, J.: 'Corner classification using Harris algorithm', *Electron. Lett.*, 2011, **47**, (9), pp. 536–538
[25] Shui, P.-L., Zhang, W.-C.: 'Corner detection and classification using anisotropic directional derivative representations', *IEEE Trans. Image Process.*, 2013, **22**, (8), pp. 3204–3218
[26] Harris, C., Stephens, M.: 'A combined corner and edge detector'. Proc. 4th Alvey Vision Conf., Manchester, UK, 1988, pp. 147–151
[27] Yang, X., Huang, Y., Li, Y.: 'An improved SUSAN corner detection algorithm based on adaptive threshold'. Proc. 2nd Int. Conf. on Singal Processing Systems, Dalian, China, July 2010, pp. 613–616
[28] Wang, Y., Li, J.: 'An improved Canny algorithm with adaptive threshold selection'. MATEC Web Conf., vol. 22, No. 01017, 2015, pp. 1–7
[29] Rong, W., Li, Z., Zhang, W., *et al.*: 'An improved Canny edge detection algorithm'. Proc. Int. Conf. on Mechatronics and Automation, Tianjin, China, August 2014, pp. 577–582
[30] Zhao, H., Xu, Z., Han, G., *et al.*: 'The fast image recognition based on grayscale features'. Proc. IEEE Int. Conf. on Computer Science and Automation Engineering, Zhangjiajie, China, May 2012, pp. 730–734