The Journal of Systems & Software 195 (2023) 111546

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

A mobile intelligent guide system for visually impaired pedestrian☆,☆☆

Wenjie Chen [a], Zimiao Xie [a], Pengxin Yuan [a], Ruolin Wang [a], Hongwei Chen [b], Bo Xiao [a,*]

[a] MoE Engineering Research Center for Software/Hardware Co-design Technology, East China Normal University, Shanghai, 200062, China
[b] Hikvision Digital Technology Co.,Ltd., Hangzhou, 310052, China

ARTICLE INFO

Article history:
Received 7 April 2022
Received in revised form 14 July 2022
Accepted 12 October 2022
Available online 27 October 2022

Keywords:
Tactile paving detection
Navigation system
Transfer learning
Visually impaired pedestrian

ABSTRACT

Traditionally, tactile walking surface indicators (TWSIs) have been used as guide tools for visually impaired pedestrians, but the bumpy bricks also bring bad experiences to other users on the road, such as the elderly, people in wheelchairs, babies in strollers and ladies wearing high heels. In this paper, we propose an intelligent guide system based on a smartphone. Videos containing information about tactile paving captured by the phone camera are processed and analyzed by the system, and guide messages are sent to the user in the form of sound or vibration. In our system, the MobileNet model fine-tuned by transfer learning is used to perform feature extraction on overlapping grids. Single Shot MultiBox Detector (SSD) is then used for TWSI detection. Finally, the user's position is determined by the Score Voting algorithm, and corresponding guide information is given. In order to further improve the real-time performance of the system, we quantize the model to compress it while ensuring accuracy. The results of experiments on real tactile paving show that our system has high accuracy and real-time performance. With our system, bumpy tactile paving bricks can be replaced with flat stickers or paint with TWSI patterns. This is comforting for other road users, and it will be easy to set up and keep up. Moreover, the types of patterns can be extended for further applications.

1. Introduction

According to the World Health Organization (WHO, 2021), at least 2.2 billion people worldwide suffer from various degrees of visual impairment. Tactile walking surface indicators (TWSIs) are now used around the world to help visually impaired pedestrians (VIPs) travel independently. The International Organization for Standardization (ISO, 2012) specifies two types of TWSIs: attention patterns and guiding patterns. As shown in Fig. 1, attention patterns consist of dozens of flat-topped domes or cones in square grids. For brevity, we describe them as dot-shaped. They are designed to call attention to hazards or decision points, and they are usually installed in the vicinity of sidewalks, stairs, ramps, etc. Guiding patterns consist of flat-topped parallel elongated bars (bar-shaped). Along the bar is the way forward. TWSIs are fixed and hard to repair and renovate. Moreover, the bumpy TWSIs cause inconvenience to other road users, especially the elderly (Ormerod et al., 2015), ladies wearing high heels, people in wheelchairs and babies in strollers.

At present, there are many solutions for tactile paving detection, which can be divided into two categories. One is based on radio frequency identification (RFID) equipment. The need for special devices and infrastructures prevents it from widespread application. The other is vision-based. It usually relies on traditional image processing methods such as edge detection and color-based region detection. Misidentification can easily occur when there are similar lines or colors on the sidewalk. In recent years, a few studies have begun to apply deep learning to this field and have achieved good results. But they only tested with historical data and did not actually deploy their systems.

In this paper, we propose a smartphone-based intelligent guide system. Videos containing information about tactile paving captured by the phone camera are processed and analyzed by the system, and guide messages are sent to the user in the form of sound or vibration. Our contributions of this system are summarized as follows:

(a) We use only a smartphone to realize the system, no other equipment is required.
(b) To enhance our model, we creatively adopt deep learning techniques from the field of tactile paving detection. In order to further compress our model, we quantize the

☆ Editor: Raffaela Mirandola.
☆☆ This paper is supported by the National Key Research and Development Program of China (2018YFB2101300), the Foundation of Shanghai Key Laboratory of Navigation and Location-Based Services, Shanghai, 200240, the National Trusted Embedded Software Engineering Technology Research Center (East China Normal University).
* Corresponding author.
E-mail addresses: wjchen@sei.ecnu.edu.cn (W. Chen), 51205902027@stu.ecnu.edu.cn (Z. Xie), 51174500150@stu.ecnu.edu.cn (P. Yuan), 51215902122@stu.ecnu.edu.cn (R. Wang), cenghao@163.com (H. Chen), bxiao@sei.ecnu.edu.cn (B. Xiao).

https://doi.org/10.1016/j.jss.2022.111546
0164-1212/

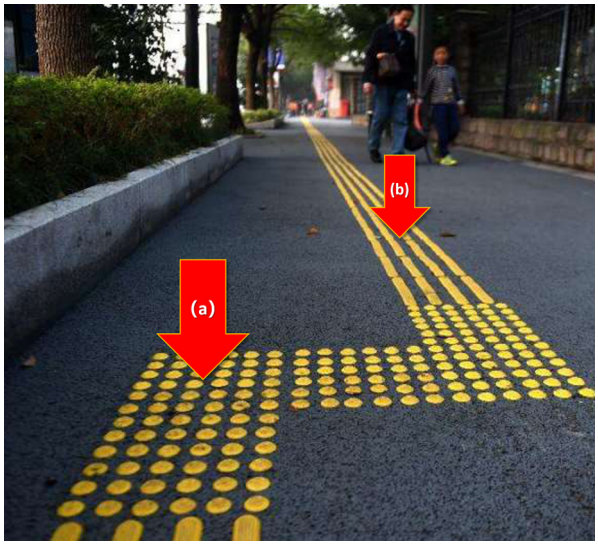

Fig. 1. The two patterns of TWSIs: (a) attention patterns (b) guiding patterns. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

lightweight network MobileNet in consideration of the constrained processing speed and storage capacity of mobile devices.

(c) We suggest the overlapped gridding method and the Score Voting algorithm to determine the user's position in relation to the tactile paving in order to remove the angle and height restrictions on the user holding the mobile device in most previous studies.

The system's image processing speed is approximately 14 FPS, and our model's accuracy is 93.76%, which can satisfy the normal usage requirements of the visually impaired.

With our system, bumpy tactile paving bricks are no longer necessary. We can directly paint patterns of tactile walking surface indicators (TWSI) on the ground instead of constructing TWSI with bricks. Even stickers printed with patterns of TWSI can be paved for temporal use. It brings three distinct benefits:

(a) Smooth roads comfort other road users, such as the elderly, people in wheelchairs, babies in strollers and ladies wearing high heels.

(b) Compared with bricks, it is easier to repair and renovate.

(c) In addition to "go straight" and "warning", more information, such as "the bus stop is on the right side", can be provided with more graphic signs.

The rest of this paper is structured as follows: Section 2 gives an overview of related concepts, methods and other background knowledge involved in this paper and previous related work. Section 3 describes the system design and methods used in the system. Section 4 shows the implementation details. Section 5 presents and analyzes the results of experiments. Section 6 conducts guide tests on both TWSIs and stickers to verify the validity of the system. Section 7 summarizes the paper.

2. Background

2.1. Related work

There are basically two categories of tactile paving detection. The first relies on RFID. The second is based on image processing.

(1) Solutions based on RFID

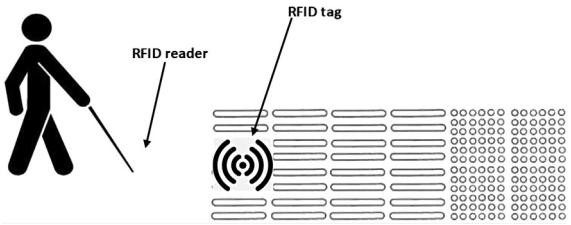

Fig. 2. Current guide method with RFID. The RFID reader is installed on the white cane, and the RFID tag is installed on the tactile paving. The RFID reader can read the information stored in the RFID tag.

Such solutions utilize RFID to obtain TWSI information (Kassim et al., 2013). Fig. 2 illustrates the RFID-based guidance system. The RFID reader is installed on the white cane, and the RFID tag is installed on the tactile paving. When the user brings the white cane close to the tactile paving, the information stored in the tag can be read. The RFID-based method requires additional equipment and facilities, which comes with a high cost, and its large-scale implementation usually needs to be led by the government.

(2) Solutions based on image processing

Typically, such solutions rely on traditional image processing methods such as edge detection and color-based region detection. Jie et al. (2010) use image segmentation in HSV color space and kirsch edge detection for tactile paving detection. Similarly, Ghilardi et al. (2016) use YCbCr color space and Laplacian edge detection. Misidentification can easily occur when there are similar lines or colors on the sidewalk. Other methods analyze the shape and texture of tactile paving bricks. However, the characteristics of tactile paving may vary by country and region. For example, the work of Jie et al. (2010) is only for yellow tactile paving in China, while (Woo et al., 2011a,b) aim to detect tactile paving in South Korea. In recent years, a few studies have begun to apply deep learning to this field, and have achieved good results. Aktaş et al. (2020) use YOLOv3 for tactile paving detection and get good accuracy and real-time performance. However, their tests are conducted using high-performance and high-cost graphics card. They do not actually deploy their system.

2.2. Object detection

Image recognition is an important field in artificial intelligence, which refers to the technology of processing, analyzing and understanding images using computers to identify targets and objects in various patterns. The four major tasks of image recognition are classification, localization, detection and segmentation (semantic segmentation and instance segmentation).

The task of image classification is to determine the category of a given image. Object localization needs to further determine the position of the object in the image (usually in the Bounding Box), and there is usually only one object. Object detection is more general than object localization. The number of objects that appear in the image is not fixed. Semantic segmentation no longer uses Bounding Box, but needs to further determine all pixels belonging to an object. Instance segmentation differs from semantic segmentation in that it needs to distinguish different instances belonging to the same category. The precision of the above four tasks increases in turn, and the difficulty of realization increases accordingly. In practical applications, a reasonable selection should be made taking into account the scenario requirements and software and hardware resources. The object detection task is performed in our system.

The current object detection methods are mainly divided into two categories: one-stage and two-stage. Two-stage means that

the detection process needs to be completed in two steps: first, candidate regions are obtained, and then they are classified. One-stage algorithm does not need to obtain candidate regions. It directly obtains the classification and position of the object. The most used two-stage methods are the Region-Convolutional Neural Network (RCNN) series, including RCNN (Girshick et al., 2014), Fast-RCNN (Girshick, 2015), Faster-RCNN (Ren et al., 2015) and Region-Fully Convolutional Network (R-FCN) (Dai et al., 2016). The most used one-stage methods are the Single Shot MultiBox Detector (SSD) algorithm (Liu et al., 2016) and the You Only Look Once (YOLO) series (Redmon et al., 2016; Redmon and Farhadi, 2017, 2018; Bochkovskiy et al., 2020). In comparison, the two-stage method has an advantage in accuracy, while the one-stage method has an advantage in speed.

2.3. Lightweight network

The system proposed in this paper is deployed on mobile device. Due to the limitations of computing power and memory, many models with high performance but complex structures and huge parameters cannot meet the requirements in terms of scale and speed, so we need to look for small and efficient networks for mobile or embedded devices. The core of the lightweight network is to lighten the network in terms of volume and speed while maintaining accuracy as much as possible.

A wide variety of lightweight networks have been proposed so far. SqueezeNet (Iandola et al., 2016) is one of the earliest proposed lightweight networks. It uses the Fire module for parameter compression. SqueezeNext (Gholami et al., 2018), as an upgraded version of SqueezeNet, adds separate convolution for improvement. Although the SqueezeNet series is not as widely used as the following ShuffleNet series (Zhang et al., 2018; Ma et al., 2018) and MobileNet series (Howard et al., 2017; Sandler et al., 2018; Howard et al., 2019), its architectural ideas are worth learning from. The ShuffleNet series is proposed by Megvii. ShuffleNetV1 (Zhang et al., 2018) uses “channel shuffle” operation to compensate for the information exchange between groups, so that the network can use pointwise group convolution to the fullest. ShuffleNetV2 (Ma et al., 2018) is designed based on five lightweight network design essentials, in which the “channel split” operation divides the input features into two parts, achieving an effect of feature reuse similar to Densely Connected Networks (DenseNet) (Huang et al., 2017). The MobileNet series was proposed by Google. MobileNetV1 (Howard et al., 2017) uses depthwise separable convolution to build lightweight networks, and the size of the model can be further controlled by two hyperparameters. The inverted residual with linear bottleneck unit innovatively integrated in MobileNetV2 (Sandler et al., 2018) improves the overall network accuracy and speed. MobileNetV3 (Howard et al., 2019) combines Automated Machine Learning (AutoML) and manual fine-tuning technology for network construction.

2.4. Transfer learning

Transfer learning refers to applying existing data or trained models to new tasks to provide reference and reduce workload. The existing data and trained models are called the source domain, and the data and models needed for the new task are called the target domain. When the required labeled data is insufficient, the existing labeled data similar to the target data can be used to expand the dataset. When the computing power is weak, the models that have been trained on large datasets can be used. These models usually already have the ability to complete some classic tasks, so it is only necessary to fine-tune the model parameters for specific scenarios.

Transfer learning can be divided into sample-based transfer, feature-based transfer, model-based transfer and relation-based transfer. Sample-based transfer reuses samples and assigns different weights to different samples according to the similarity of samples, so as to improve the effectiveness of transfer. Feature-based transfer transforms features spatially to make them similar in source and target domains, so that the same feature extraction model can be used for feature extraction. Model-based transfer means that the target domain shares the model parameters of the source domain or fine-tunes model parameters on the basis of the source domain. Relation-based transfer refers to mining the relationships in the source and target domains and performing analogical transfer. This type of transfer learning is less used. Due to the lack of tactile paving datasets, we adopt model-based transfer to fine-tune the parameters of the trained model using the dataset collected by ourselves.

2.5. Model compression

The research on object detection algorithms mainly has two aspects: (1) Focusing on complex deep learning models, and pursue higher performance; (2) Focusing on actual deployment of models, and pursue algorithm stability, efficiency, real-time performance and low energy consumption. The system proposed in this paper is deployed on the mobile terminal. Based on the characteristics of mobile devices, real-time performance and power consumption need to be considered (Krishnamoorthi, 2018), so the model needs to be compressed.

Compression of deep learning models can be done in two ways: (1) Modify the network structure to obtain a network with less computation or model parameters, mainly including methods such as grouped convolution, convolution decomposition, and neural structure search. The MobileNet model used in this paper adopts the network structure compression method of convolution decomposition; (2) Parameter quantization, by performing an affine transformation on the weight or activation function, transforms the model parameters into discrete, low-precision numerical points. In this paper, INT8 quantization is used to convert the precision of the convolution operation from 32-bit floating-point numbers to 8-bit integers. In theory, the size of the model becomes 1/4 of the original, and the amount of calculation is greatly reduced.

3. Method

The overall design of the system is shown in Fig. 3, which mainly includes four modules. This section presents the most critical technologies and methods of the system.

3.1. System design

As shown in Fig. 3, the input of the system is the video shot by the phone camera, and the output of the system is the guide information. The system is divided into four modules, namely the Video Processing module, the Object Detection module, the Results Analysis module and the Data Upload module. First, the input video is preprocessed by the Video Processing module, which is responsible for intercepting the video frame and adjusting it to an appropriate size. Since the system model only accepts byte streams as input, this module also needs to serialize images. Then the Object Detection module performs TWSI detection. We use the MobileNetV2-SSD model in this module, where MobileNetV2 is used for feature extraction, and SSD is used for object detection. To adapt the model to the TWSI detection task, we employ transfer learning to fine-tune the model using our own TWSI datasets. Next, the results from the Object Detection

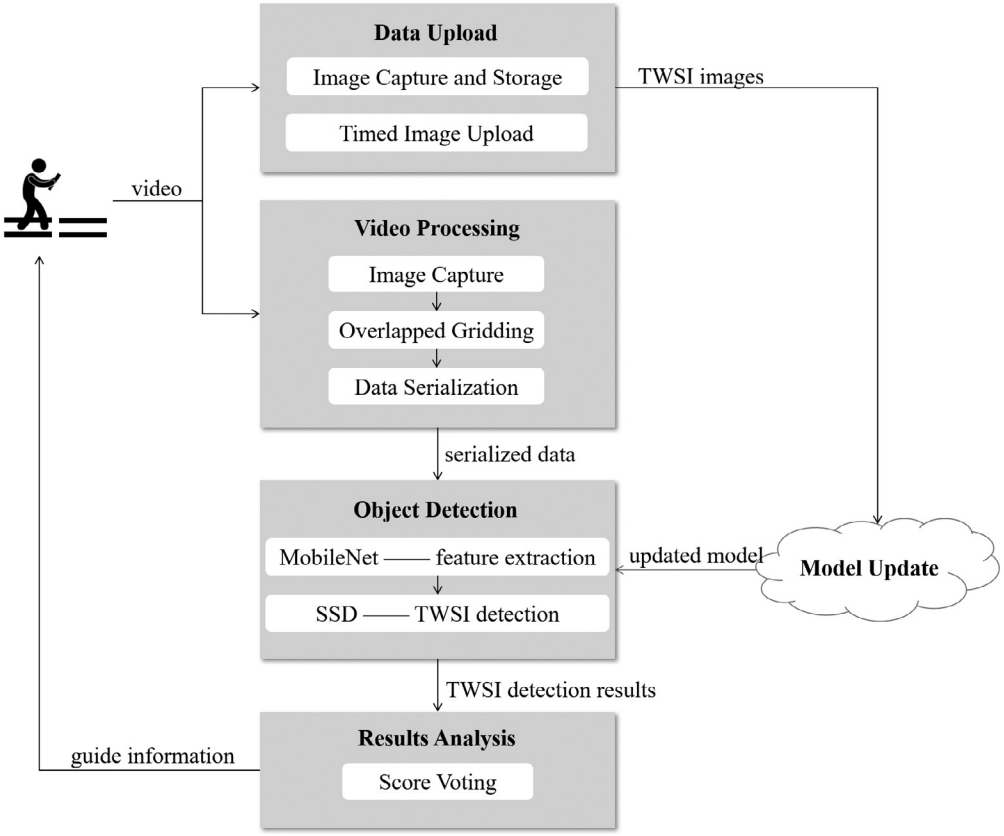

Fig. 3. Our guide system.

module are analyzed by the Results Analysis module to obtain the user's position, and the corresponding guide information is sent to the user in the form of sound or vibration. Finally, considering the limited amount of data, the system will automatically capture video frames for storage, and then upload them to the cloud server regularly to supplement the existing dataset. This work is performed by the Data Upload module.

With our system, only the visual patterns of "bar-shaped" and "dot-shaped" are essential for discrimination. Tactile surfaces (raised or sunken) are not necessary any more. Therefore, we can replace the current bumpy TWSIs with simple flatten stickers or paint with TWSI patterns. Fig. 4 shows the TWSI bricks and our stickers. What is more, we can use more patterns for additional information. For example, a special pattern represents "a crossroad ahead". It is more flexible and efficient than traditional TWSIs.

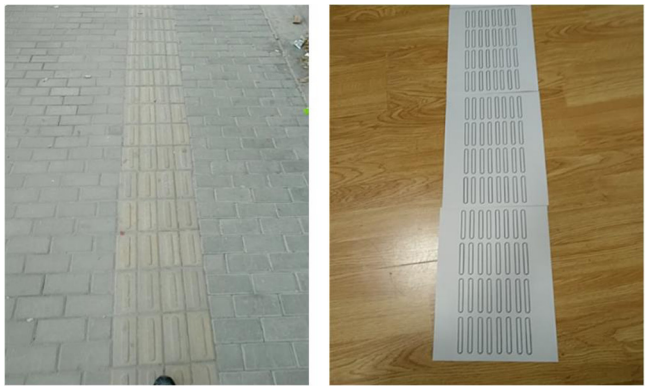

Fig. 4. TWSIs and sticker. With our system, simple flatten stickers or painting with TWSI patterns can replace bumpy TWSIs.

3.2. Dataset

As far as we know, there is no open dataset of TWSIs. We set up the dataset by ourselves. The images are partially captured from the camera and smartphone by ourselves and partially copied from the Internet. The images in the dataset are divided into "with TWSIs (T)" images and "without TWSIs (NT)" images. Furthermore, *T* images are divided into "bar tiles" (BT) and "dot tiles" (DT). In our dataset, the number of BT is 454, DT is 385, and NT is 484. Fig. 5 shows some typical images. For our system, 80% of them are used as training sets and the remaining 20% are test sets.

Fig. 5. Partial of our dataset. There are three categories in the dataset, namely BT (images that contain bar tiles), DT (images that contain dot tiles) and NT (images that do not contain TWSIs).

Our dataset has 1323 images, which contain dozens of scenes. In this paper, all scenes are divided into indoor scenes and

outdoor scenes according to light intensity. What is more, different heights and angles are recorded, so our system is insensitive to height and angle differences. We have demonstrated it experimentally.

3.3. System model and transfer learning

The application of convolutional neural networks (CNN) in the field of computer vision has achieved very good results, but the cost is the high complexity of models. For mobile devices with low computing power and small storage space, such a huge and complex model is difficult to deploy and apply. Moreover, for navigation systems, in addition to pursuing high accuracy, real-time performance must also be considered. Therefore, choosing a small and efficient model is crucial. MobileNet reduces model size and computational complexity by applying depthwise separable convolutions. It can achieve a good balance between accuracy and latency under the resource constraints of mobile devices. Therefore, in this paper, the MobileNet model is used to extract features from images, and on this basis, the SSD algorithm is used for TWSI detection.

A new machine learning model cannot get satisfactory training results when there is not enough valuable data, which is called "cold start". The datasets used in this paper are all captured by ourselves or collected from the Internet. It is difficult to ensure the scale and quality of the dataset. Using random parameters to initialize the model, that is, cold start, can easily lead to overfitting problems. We solve this problem by transfer learning. The process of transfer learning is shown in Fig. 6. First, we utilize the ImageNet dataset (Deng et al., 2009) to train the MobileNet model. Many current studies have obtained a good feature extraction model by training on ImageNet (Simonyan and Zisserman, 2014; He et al., 2016). Then, we fine-tune the model with the TWSI dataset we created.

3.4. Overlapped gridding

Many of the previously presented studies have limitations on the height and angle at which users can hold the camera. For example, Ghilardi et al. (2016) assumed that all images were taken at an average height of 1.50 m and an angle measuring 45 degrees relative to the ground. The premise of the method proposed by Jie et al. (2010) is that the Personal Digital Assistant (PDA) is kept parallel to the ground. This greatly increases the inconvenience for users.

In order to reduce the influence of the height and angle of users holding the camera and fully utilize all the information contained in different parts of images, we divide each image into 9 grids. The grids are processed one by one and synthesized finally. In this way, we comprehensively exploit information of all grids of the input image. That means the information of different directions of pavement can be considered and the effect of noise on every section is depressed, which improves the robustness of the system.

On the contrary, Ghilardi et al. (2016) divided images into 60–80 grids to get the outline of TWSIs. We repeated the experiment on a smartphone and found that each grid was too small to recognize. It is not suitable for our situation. In our application, the only thing that needs to be determined is where to go next (left, right, or straight) from the user's perspective. The fine-grained outline is not necessary.

In this paper, we use 9 grids to make full use of the global information of images. The image in Fig. 7 is a video frame used in the experiment. After division, the system recognizes the type of each grid independently. The recognition result is that grid 2, 3, 8, and 9 are "bar-shaped"; grid 4, 5, and 6 are "dot-shaped"; and grid 1, 7 are none-TWSIs. Based on this, we know the pedestrian is on the left of the pavement and needs to move a little to the right at the next step.

The gridding method above has a clear separating boundary, which may lose the feature information on the boundaries. As an improvement, we use overlapped gridding instead. As shown in Fig. 8, each grid occupies a quarter of the whole frame. The information of overlapping part will be calculated twice. Compared with the non-overlapped gridding method, the overlapping method has a larger receptive field, so it can better extract the feature information in each direction and obtain more accurate pavement information.

3.5. Score Voting

After each grid is recognized, the results of nine grids representing different directions of the pavement are sequentially sent to Tensorflow Lite for further analysis. Then the Score Voting rule is used to conclude the position of the pedestrian.

To clearly separate these three classes, we set the score of BT to 5, DT to 100 and NT to 0. Therefore, if the total score is over 100, it means that there is at least one DT. On the contrary, score 0 means that there is no TWSI found in the whole image. Algorithm 1 shows the Score Voting algorithm.

Algorithm 1 Score Voting

Require: the detection results of 9 grids
Ensure: guide message
if *grid is NT* **then**
  grid_score = 0
else if *grid is BT* **then**
  grid_score = 5
else if *grid is DT* **then**
  grid_score = 100
end if
$l\_score \leftarrow \text{SUM}(grid[0], grid[1], grid[2])$
$m\_score \leftarrow \text{SUM}(grid[3], grid[4], grid[5])$
$r\_score \leftarrow \text{SUM}(grid[6], grid[7], grid[8])$
if $l\_score + m\_score + r\_score \leq 15$ **then**
  return *Nothing detect, go to pavement first*!
else if $l\_score + m\_score + r\_score \geq 300$ **then**
  return *ATTENTION! Dot tile.*
else if $l\_score \geq 10 \wedge r\_score < 5$ **then**
  return *please go left a little*!
else if $r\_score \geq 10 \wedge l\_score < 5$ **then**
  return *please go right a little*!
else if $m\_score \geq 10$ **then**
  return *please keep straight*!
end if

We first allocate a score to each grid according to the recognition results. Then we use the sum of the left three grids' scores as the total result of the left side of the pavement. So do the middle three grids and the right three grids. Finally, the three scores of left, middle and right are integrated to obtain the overall result of the pavement.

4. Implementation

For mobile device applications, model compression and deployment are critical. This section describes how we use transfer learning to train the model and how to compress and deploy the model.

4.1. Model training using transfer learning

We use Tensorflow for transfer learning and configure the model according to the TWSI detection task. The number of categories is set to 2 ("bar-shaped" and "dot-shaped"). When searching for candidate areas, IoU $>$ 0.5 is a correct match. In the TWSI task, we need to determine the category and region of

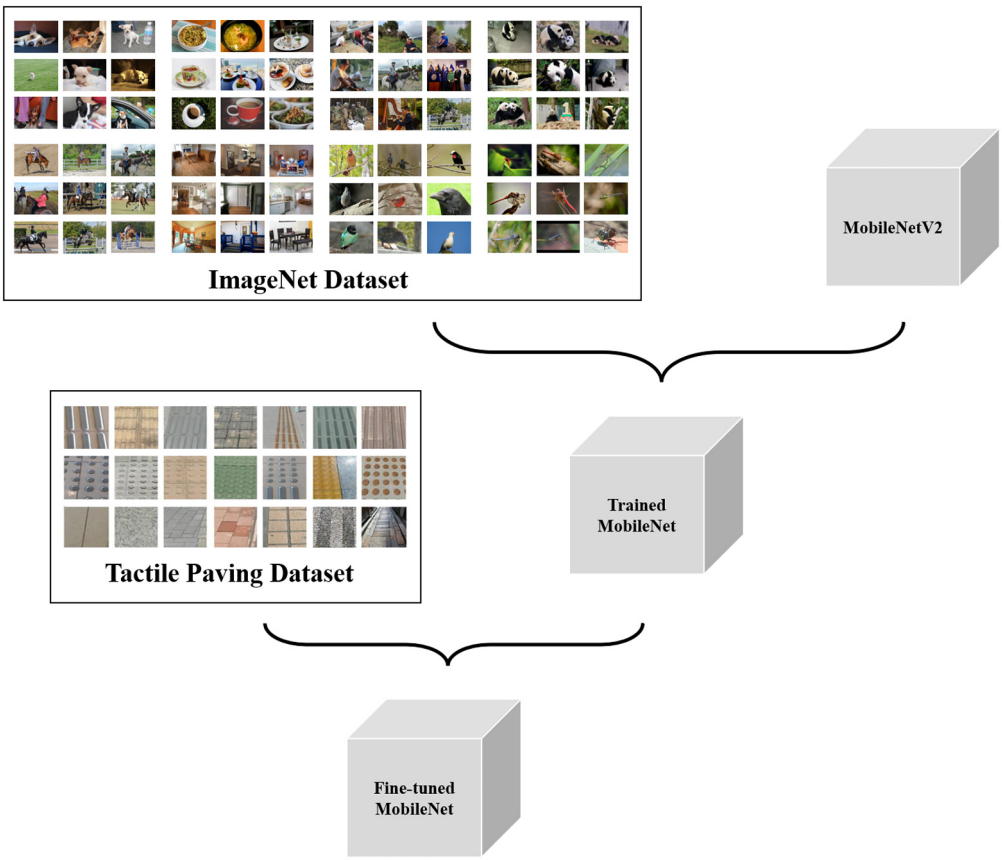

Fig. 6. Transfer learning method.

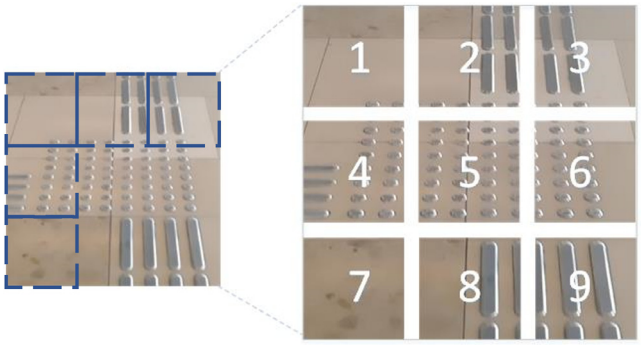

Fig. 7. Image gridding. This gridding method has a clear separating boundary, which may lose the feature information on the boundaries.

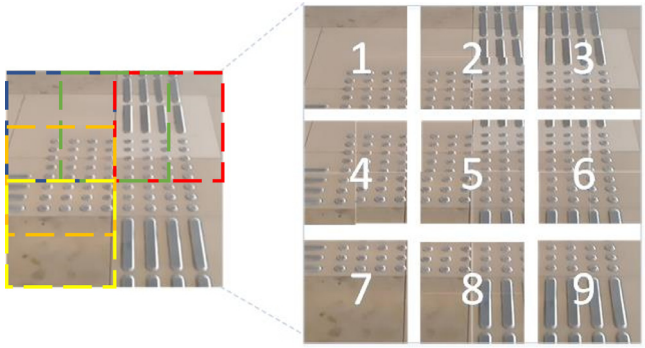

Fig. 8. Overlapped gridding. Each different colored dashed box represents a different grid. Each grid occupies a quarter of the whole frame. The information of overlapping part will be calculated twice which helps obtain more accurate pavement information. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

an object, so the loss function is composed of two parts. The loss function corresponding to the category is the cross-entropy loss function, and the region is the L1 regression loss function. The ratio of the number of positive and negative sample candidate regions is set to 1: 3, that is, for each positive sample, up to 3 negative samples can be reserved. We calibrate categories and corresponding candidate regions to form a record file and process object labels to form a pbtxt file.

During the fine-tuning stage, parameters for feature extraction are frozen, and parameters for classification in the last layer of the model are randomly initialized with a Gaussian distribution (mean 0, variance 0.001). The model is optimized using the Adam method with a learning rate of 0.0005 and a cross-entropy loss function. We use 1000 epochs for fine-tuning, and the accuracy stabilizes at around 95% after 800 epochs. The accuracy and cross-entropy changes during model training are shown in Fig. 9.

4.2. Model compression and deployment

The memory and computing resources of mobile devices are limited. In order to better deploy the system on mobile devices, model quantization is used to convert model parameters from 32-bit floating point numbers to 8-bit integers. Theoretically, the size of the model will be reduced to 1/4 of the original, and the amount of computation will be greatly reduced.

In order to determine the most suitable threshold for quantization and reduce the accuracy loss of the quantized model

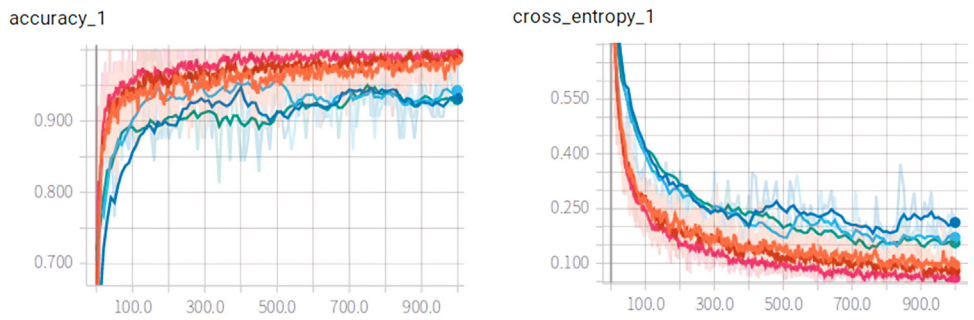

Fig. 9. The accuracy and cross-entropy changes during model training.

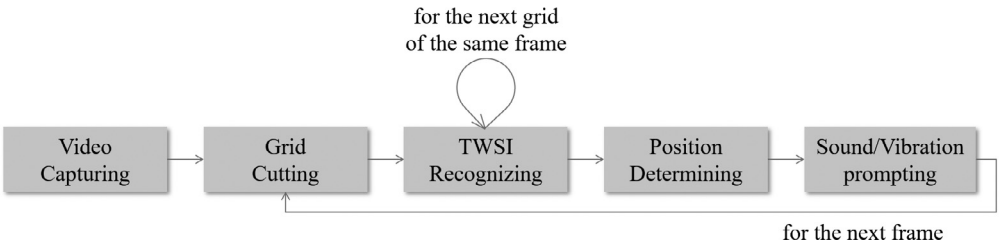

Fig. 10. The workflow of our system on smartphone.

as much as possible, we consider the prior knowledge of the model, that is, the distribution of activations for each layer of the model when training or inference with 32-bit floating point precision. The specific method is as follows: (1) First, select a subset of the validation set as the calibration set, and the data in the calibration set should be diverse and representative. (2) Then perform inference with 32-bit floating point precision on the calibration set. Draw the activation value histogram of each layer of the network and group these values. Traverse the middle value of each group, and select the activation value that minimizes KL divergence. (3) After completing step 2 for each layer, return a series of activation values (one activation value for each layer) to create a calibration parameter table. (4) Each layer is quantitatively calibrated according to the calibration parameter table.

To be efficient, the training and fine-tuning of the model are performed on a PC. The PC model cannot be directly applied to smartphones. Its format needs to be conversed by Tensorflow Lite, a lightweight solution for mobile and embedded devices. We deploy Tensorflow Lite on an Android smartphone.

4.3. Workflow on smartphone

Our system is implemented on a smartphone as an Android application. The workflow on a smartphone is shown in Fig. 10. The mobile phone camera obtains real-time videos during the user's walking, and the system processes videos frame by frame. First, each frame is divided into 9 overlapping grids that represent different directions of pavement. Second, the system model is called to identify and locate TWSI in each grid. Third, a comprehensive analysis is conducted. Finally, the guide information is generated and sent to the user in the form of sound or vibration.

5. Results

The mobile device used in this paper is based on the Android operating system of version 7.1.1 and is equipped with a Snapdragon 660 processor with a main frequency of 2.2 GHz. The running memory size is 6 GB.

We conduct experiments on the fine-tuned model using the dataset we created. For the fine-tuning MobileNet model with transfer learning, we combine the pre-trained parameters with the ImageNet dataset and fine-tune only the parameters of the final classification layer. Thanks to transfer learning, we can use a well-trained feature extraction layer, and the fine-tuning stage will be faster than the training stage.

5.1. Feature extraction capability of the model

We use the MobileNetV2-SSD model for TWSI detection, where MobileNetV2 is used to extract features from input images to obtain feature maps at different levels. Then, SSD uses feature maps to determine the categories and positions of objects. Therefore, the ability of MobileNetV2 network to extract features directly affects the final effect of the model.

To evaluate the ability of the feature extraction of the model we used, we compared it with the work of Ghilardi et al. (2016), which aimed to classify images into two categories — TWSIs and non-TWSIs. Ghilardi's dataset is not the same as ours. For a fair comparison, we used the same dataset as Ghilardi's and fine-tuned our model with only 10% of their dataset. We used the same evaluation metrics: Accuracy to calculate the accuracy of classification; Sensitivity to calculate the model recall rate; Specificity to calculate the true negative rate.

Ghilardi's dataset includes 561 images, 360 of which include TWSIs, and the rest do not. After the fine-tuning process, we evaluated all 561 images on our fine-tuned MobileNet model and got 23 misclassifications for TWSIs and 12 misclassifications for non-TWSIs. The model evaluation results are shown in Table 1. Compared with Ghilardi's work, our model has a great improvement in accuracy and recall, and specificity has also improved. The feature extraction model based on transfer learning adopted in this paper can achieve satisfactory accuracy and has good generalization ability, which provides a good basis for the subsequent TWSI detection task.

5.2. Influence of expansion factor on model performance

The MobileNet model is based on depthwise separable convolutions, which greatly reduces computation and model size. Before the depthwise separable convolution, a $1 * 1$ convolution

Table 1
Comparison with Ghilardi's work on their dataset.

		Ghilardi's	Ours
Accuracy	(TP+TN)/ (P+N)	88.48%	93.76%
Sensitivity	TP/P	85.31%	93.61%
Specificity	TN/N	93.53%	94.03%

layer is added in MobileNetV2 to improve the dimension of feature maps and increase the expressiveness of the model. The ratio by which the dimension is boosted is called the expansion factor. Different expansion factors result in different parameter sizes and computational complexity. The smaller the expansion factor is, the smaller the parameters' size and computational complexity of the model are, but the capability of feature extraction may be reduced. We analyze the ability of feature extraction of MobileNetV2 with expansion factors of 12, 8, 6, and 3 respectively. The accuracy, latency and model size of the model under different expansion factors are shown in Table 2.

Latency is measured on a smartphone. It is the time cost of recognition per grid. Model size is the size of the fine-tuned model on the PC. The TFLite size is the final size of the model, which is converted to an Android-compatible format. From Table 2, we can see that as the expansion factor decreases, the accuracy of the model decreases slightly (but remains above 93%), but model size and latency decrease significantly. When the expansion factor is 12, the latency and TFLite size of the model are 225 ms/grid and 16.9 M, respectively. When the expansion factor is 3, the two values become 25 ms/grid and 1.9M, respectively. Therefore, we choose an expansion factor of 3 to deploy our system.

5.3. The effect of model quantization

In this part, we evaluate the effect of model quantization and the accuracy loss caused by quantization in terms of model size and latency. After fine-tuning the model in the cloud, the model will be deployed on mobile devices, so the performance analysis will be performed on mobile devices. In order to evaluate the performance of the quantized model, we deploy the 32-bit floating-point model and the 8-bit integer model on the same smartphone, respectively. By testing the two models in real scenarios, we compare and analyze the impact of model quantization on real-time performance.

The results are shown in Fig. 11. MobileNetV2 (INT8) means that only the MobileNetV2 module is used for image classification, and the classification result is BT, DT or NT. The model has been quantified. MobileNetV2 + SSD (INT8) represents the task of TWSI detection using the MobileNetV2 module and SSD module. The amount of computation is greater than that of using only MobileNetV2. The model has been quantized. MobileNetV2 + SSD (FP32) represents the task of TWSI detection using the MobileNetV2 module and the SSD module. The model is 32-bit floating point precision, as a comparison with the quantized model. From the results of the experiment, it can be seen that the inference of the quantized model is accelerated by about 3.8 times.

6. Evaluation for different user environments

6.1. Indoor — subway station

The subway station is an indoor scene where guide services are used more often, so we choose it as one of the indoor test scenes. We walked along the tactile paving with the smartphone in hand, and the system's real-time detection results are shown

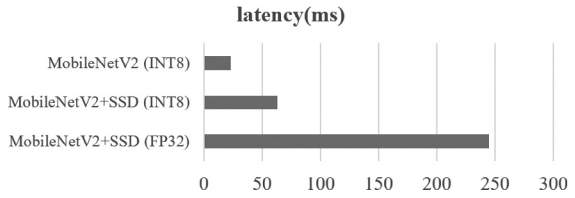

Fig. 11. The latency of models with different precision and compositions.

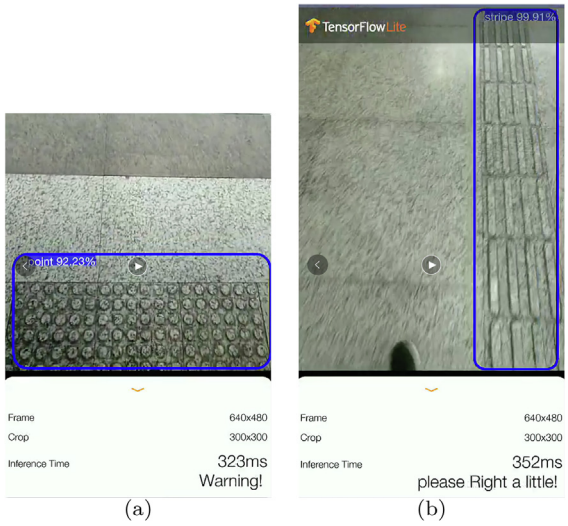

Fig. 12. The results of test in subway station (FP32). (a) The system detected dot-shaped TWSIs. (b) The system detected bar-shaped TWSIs. The latency is about 330 ms.

in Fig. 12. In the picture on the left, the system detected dot-shaped TWSIs. Its function is to remind VIPs to pay attention to turning or danger. In the picture on the right, the system detected bar-shaped TWSIs. As can be seen from Fig. 12, the tactile paving was on the right side of the user, and our system gave the corresponding guide information, prompting the user to move to the right.

The model used in the above experiment has 32-bit floating point precision. It can be seen that the latency is about 330 ms. When we adopt the quantized model, the latency is greatly reduced, as shown in Fig. 13.

As can be seen from Fig. 13, the system can still provide correct guide information, and the latency is significantly reduced, at 60–80 ms. Compared to the model with 32-bit floating-point parameters, the inference speed of the quantized model is 4–5 times faster.

6.2. Outdoor — sidewalk

We chose the sidewalk as one of the outdoor test scenarios. Outdoor tactile paving has many disturbances, such as abrasion and shading, which may cause the effect of recognition to decline. As in the indoor scene test, we walked along the tactile paving with the smartphone in hand. The real-time detection results are shown in Fig. 14. In the left picture, the system detected bar-shaped TWSIs. It could be seen that the user deviated from the tactile paving. Our system correctly identified the location of the tactile paving and prompted the user to move to the left. Due to the shade, the recognition probability was only 46.09%. In the right picture, there were both bar-shaped TWSIs and dot-shaped TWSIs, but due to the wear and tear of the tactile paving, the

Table 2
The accuracy, latency and size of the model with different expansion factors.

Width multiplier	Accuracy	Latency (ms/grid)	Model size (M)	TFLite size (M)
1.0	0.9837	225	17.1	16.9
0.75	0.9638	133	10.5	10.3
0.50	0.9366	70	5.5	5.3
0.25	0.9375	25	2.0	1.9

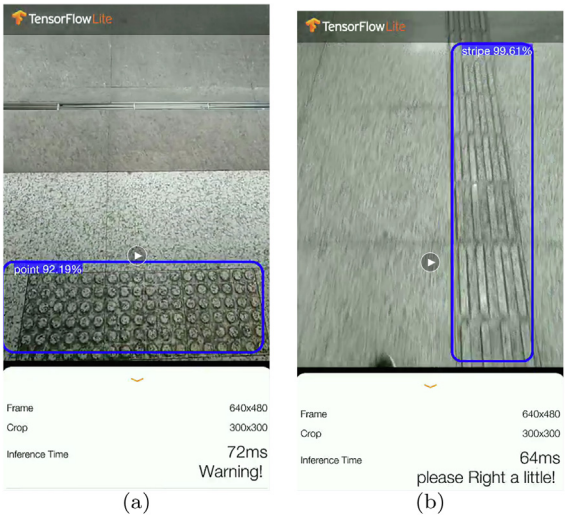

Fig. 13. The results of test in subway station (INT8). (a) The system detected dot-shaped TWSIs. (b) The system detected bar-shaped TWSIs. The latency is about 60–80 ms.

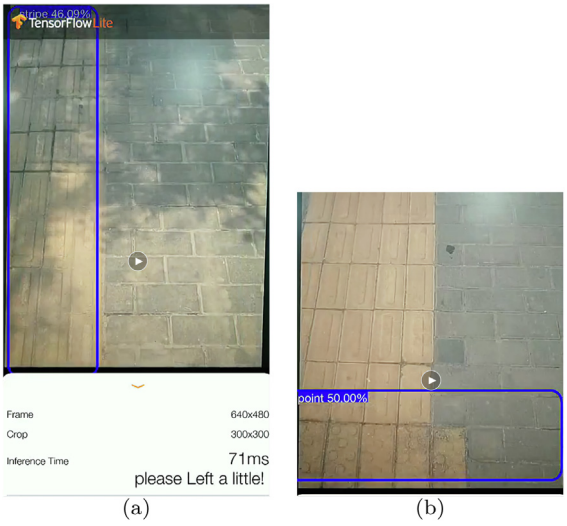

Fig. 14. The results of test on sidewalk. (a) The system detected bar-shaped TWSIs. Due to the shade, the recognition probability was only 46.09%. (b) Due to the wear and tear of the tactile paving, the system recognized the dot-shaped TWSIs with a probability of 50.00%, and did not recognize the bar-shaped TWSIs.

system recognized the dot-shaped TWSIs with a probability of 50.00%, and did not recognize the bar-shaped TWSIs.

The model used in the above experiment has been quantized, and the inference time for a single image is 71 ms, that is, 14 FPS, which meets the real-time requirements.

6.3. Stickers

The intelligent blind guidance system designed in this paper is suitable for both traditional bumpy bricks and flatten pictures.

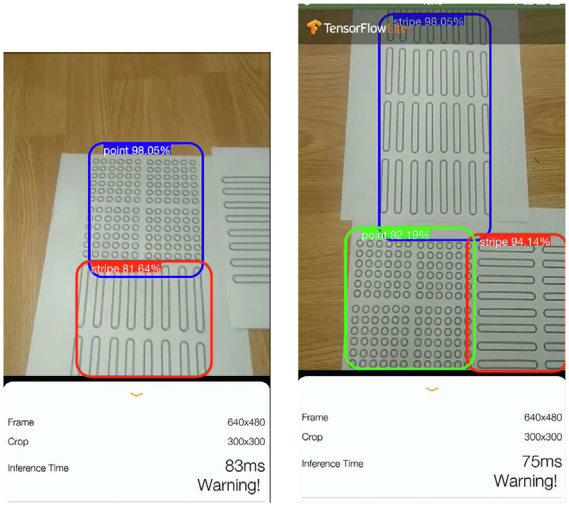

Fig. 15. The results of test on sidewalk. The system can accurately identify two types of TWSIs.

Bumpy bricks have problems such as easy damage and inconvenience to change. These problems can be avoided very well by using flatten stickers or painting with TWSI patterns. We used the printed images of TWSI to replace bricks and tested them indoors. As shown in Fig. 15, the system can accurately identify two types of TWSIs.

7. Conclusion

In this paper, we propose an intelligent guide system for visually impaired pedestrians based on smartphones. We use smartphones as identification and analysis tools for the system. Our recognition module is based on the MobileNet model fine-tuned by transfer learning and the SSD object detection algorithm. The analysis module uses Overlapped Gridding and Score Voting to obtain comprehensive guide information. In order to further improve the real-time performance of the system, we perform INT8 quantization on the model. Due to the lack of tactile paving datasets, we create a TWSI dataset ourselves, and then fine-tune the MobileNet model on our dataset. After that, we deploy the model on an Android smartphone via Tensorflow Lite. Finally, we test our system in real-life walking scenarios, and the system is able to achieve high accuracy and real-time guide on both real tactile paving and stickers.

The source code and video demonstrations of the system is available at github.com/zimiao1229/intelligent-guide-system.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Aktaş, A., Doğan, B., Demir, Ö., 2020. Tactile paving surface detection with deep learning methods. J. Fac. Eng. Archit. Gazi Univ. 35 (3), 1685–1700.

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M., 2020. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

Dai, J., Li, Y., He, K., Sun, J., 2016. R-fcn: Object detection via region-based fully convolutional networks. Adv. Neural Inf. Process. Syst. 29.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. Ieee, pp. 248–255.

Ghilardi, M.C., Macedo, R.C., Manssour, I.H., 2016. A new approach for automatic detection of tactile paving surfaces in sidewalks. Procedia Comput. Sci. 80, 662–672.

Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., Zhao, S., Keutzer, K., 2018. Squeezenext: Hardware-aware neural network design. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops pp. 1638–1647.

Girshick, R., 2015. Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 580–587.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al., 2019. Searching for mobilenetv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1314–1324.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4700–4708.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K., 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.

ISO, 2012. ISO 23599:2012 assistive products for blind and vision-impaired persons — Tactile walking surface indicators. https://www.iso.org/standard/55867.html, Accessed Mar 15, 2022, [OL].

Jie, X., Xiaochi, W., Zhigang, F., 2010. Research and implementation of blind sidewalk detection in portable eta system. In: 2010 International Forum on Information Technology and Applications. 2, IEEE, pp. 431–434.

Kassim, A., Jaafar, H., Azam, M., Abas, N., Yasuno, T., 2013. Design and development of navigation system by using RFID technology. In: 2013 IEEE 3rd International Conference on System Engineering and Technology. IEEE, pp. 258–262.

Krishnamoorthi, R., 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. Ssd: Single shot multibox detector. In: European Conference on Computer Vision. Springer, pp. 21–37.

Ma, N., Zhang, X., Zheng, H.-T., Sun, J., 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 116–131.

Ormerod, M., Newton, R., Maclennan, H., Faruk, M., Thies, S., Kenney, L., Howard, D., Nester, C., 2015. Older people's experiences of using tactile paving. Munic. Eng. 168 (me1), 3–10.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 779–788.

Redmon, J., Farhadi, A., 2017. YOLO9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7263–7271.

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. 28.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

WHO, 2021. Blindness-and-visually-impairment. https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment, Accessed Mar 15, 2022, [OL].

Woo, B.-S., Yang, S.-M., Jo, K.-H., 2011a. Brick path detection from shape pattern and texture feature. In: 2011 IEEE/SICE International Symposium on System Integration (SII). IEEE, pp. 78–83.

Woo, B.-S., Yang, S.-M., Vavilin, A., Jo, K.-H., 2011b. Brickpath region detection using color and shape pattern information. In: Proceedings of 2011 6th International Forum on Strategic Technology, Vol. 2. IEEE, pp. 720–724.

Zhang, X., Zhou, X., Lin, M., Sun, J., 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6848–6856.

Wenjie Chen received the B.Eng. (1998) degree in electronic engineering from Zhejiang University (China) and the Ph.D. degree (2007) in computer science from Fudan University (China). He is currently an associate professor in the School of Software Engineering, East China Normal University, P.R. China. His research interests include intelligent systems and hardware/software co-design.

Zimiao Xie currently studying for a master's degree at East China Normal University, majoring in software engineering. Her main research direction is artificial intelligence and image processing.

Pengxin Yuan received a master's degree in software engineering from East China Normal University. His main research direction is image processing and hardware/software co-design.

Ruolin Wang currently studying for a master's degree at East China Normal University, majoring in software engineering. Her main research direction is image processing and embedded systems.

Hongwei Chen currently working in Hikvision Digital Technology Co., Ltd. He has been deeply involved in the field of computer vision for a long time.

Bo Xiao received the Ph.D. degree in Information and Communication Systems from Shanghai Jiaotong University, China, in 1999. He is an Associate Professor with the School of Software Engineering, East China Normal University, China. His research interests include IoT, ad hoc networks and machine learning.