The Journal of Supercomputing (2021) 77:7737–7756
https://doi.org/10.1007/s11227-020-03568-5

CamNav: a computer-vision indoor navigation system

Abdel Ghani Karkar[1] · **Somaya Al-Maadeed**[1] · **Jayakanth Kunhoth**[1] · **Ahmed Bouridane**[2]

Accepted: 14 December 2020 / Published online: 8 January 2021

Abstract
We present CamNav, a vision-based navigation system that provides users with indoor navigation services. CamNav captures images in real time while the user is walking to recognize their current location. It does not require any installation of indoor localization devices. In this paper, we describe the techniques of our system that improve the recognition accuracy of an existing system that uses oriented FAST and rotated BRIEF (ORB) as part of its location-matching procedure. We employ multiscale local binary pattern (MSLBP) features to recognize places. We implement CamNav and conduct required experiments to compare the obtained accuracy when using ORB, the scale-invariant feature transform (SIFT), MSLBP features, and the combination of both ORB and SIFT features with MSLBP. A dataset composed of 42 classes was constructed for assessment. Each class contains 100 pictures designed for training one location and 24 pictures dedicated for testing. The evaluation results demonstrate that the place recognition accuracy while using MSLBP features is better than the accuracy when using SIFT features. The accuracy when using SIFT, MSLBP, and ORB features is 88.19%, 91.27%, and 96.33%, respectively. The overall accuracy of recognizing places increased to 93.55% and 97.52% after integrating MSLBP with SIFT with ORB, respectively.

Keywords Indoor navigation · Mobile technology · Multiscale local binary pattern features · ORB features · Scene recognition · SIFT features · SVM · Deep learning

✉ Abdel Ghani Karkar
a.karkar@qu.edu.qa

Somaya Al-Maadeed
s_alali@qu.edu.qa

Jayakanth Kunhoth
j.kunhoth@qu.edu.qa

Ahmed Bouridane
ahmed.bouridane@northumbria.ac.uk

[1] Department of Computer Science and Engineering, Qatar University, Doha, Qatar

[2] Department of Computer and Information Sciences, University of Northumbria, Newcastle upon Tyne, UK

1 Introduction

Indoor navigation systems are designed to provide routing guidance for people inside buildings or covered areas where global positioning system (GPS) signals are not reachable. Indoor navigation systems often use localization devices that use radio waves, beacons signals, acoustic signals, or other signals. Diverse indoor navigation systems have been implemented and made available for commercial use. In addition, some of these systems have been developed for mobile handheld devices [10, 24, 38]. In fact, the advances in mobile technology opened new horizons in growing computing environments [29]. With the availability and ease of mobile devices, people can have prompt access to information, as they can put their mobile devices in their pockets, around their necks, or in their hands. With the current capability of mobile devices, users can track their emotional states, monitor their behaviors, capture high-quality detailed images, and perform complex data processing tasks. Computer vision applications have demonstrated their effectiveness in promoting the use of mobile applications. A recent computer-vision indoor navigation system has been proposed to provide required indoor navigation services [43]. However, the proposed system, namely Travi-Navi, has some common limitations when using computer vision techniques:

1. The oriented FAST and rotated BRIEF (ORB) algorithm has been employed for processing pictures for navigational guidance. It has been indicated that ORB is more efficient than SIFT [30]. However, pictures that either do not have or have a limited number of ORB features may result in incorrect place recognition.
2. The system trains [43], using ORB features, a support vector machine (SVM) model [33] to predict locations. However, the employment of local binary pattern (LBP) features can enhance the prediction results in both SVM and deep learning models.
3. Images are processed on the mobile device, which dramatically reduces the battery life of the device. On-server processing can improve the life time of the battery.

From a research perspective, the use of SIFT has demonstrated its capability to perform image-matching better in some scenarios [17]. For this reason, SIFT is considered as part of the assessment. In addition, the use of deep learning (DL) has achieved considerable practical successes and has been widely applied in the field of computer vision [19]. In effect, DL enables computational models that are formed from several layers to learn the representations of data with diverse levels of abstraction. It is able to discover complex structures in massive datasets by employing back-propagation algorithms to identify how a machine could improve training [31]. For this reason, the employment of deep learning concepts to recognize locations using extracted features is also considered. In this paper, we propose a computer-vision mobile-based navigation system, which we call CamNav. It provides users with indoor navigation services such as place recognition and route guidance. To concurrently inform the users about their location, we configured the system to capture images and process them in real time as in [43]. We propose the use of multiscale local binary pattern features (MSLBP) [28, 26] to overcome the aforementioned limitations, and we make our system capable of performing the processing tasks on a server. In addition, we created a convolutional neural network (CNN) model and trained it using our constructed dataset with extracted features. The dataset is composed of 42 classes in which each class represents one actual location in a building. When using the mobile application, the relevant coordinates will be obtained to update the location of the user. Figure 1 shows the overall architecture of the system. It is composed of five components: (1) an image dataset, (2) a support vector machine (SVM) model, (3) a (DL) model, (4) an image processing component, and (5) an end-user mobile application. To collect the required images, we configured the end-user mobile application to run in *picture collection* mode. In this mode, the mobile application will capture images continuously and send them to the server. We revised the received images and added required descriptive files such as location info and point on map. Descriptive files are stored in JSON format in each directory next to the images, as shown in Fig. 2. We compare the performance of our system with the computer vision technique that is developed in recently developed systems (Travi-Navi [43] and iNavigation [39]) by comparing the usage of ORB, SIFT and MSLBP features. Moreover, we combine the ORB and SIFT features with the MSLBP features to improve the results. In addition, we create a DL model and train it with the extracted features. We conduct the required statistical analysis to demonstrate the effectiveness of our approach.

The remaining sections of the paper are organized as follows. In Sect. 2, we detail relevant related work. In Sect. 3, we describe our system and its features. We evaluate the system in Sect. 4. Finally, in Sect. 5, we conclude the paper.

2 Related work

Diverse systems have been proposed in recent decades to provide people with indoor navigation services. These systems vary from simple systems that can be used directly on mobile devices to complex systems that require necessary additional devices [2, 14–16]. In this section, we focus on a computer vision-based indoor navigational system and review relevant related work.

Deniz et al. [9] proposed an indoor localization method by integrating the edge detection mechanism with text recognition. The edge detection mechanism used a Canny edge detector [5] to detect the edges of images. However, the employment of edge detection in places that have a limited number of edges will result in incorrect place recognition. Tian et al. [37] proposed a navigation system for people with visual impairments (VI). The proposed system consists of a door-detection and a text-recognition module. Door detection is achieved by utilizing a Canny edge detector along with a curvature-based corner detector. The text recognition module uses mean shift-based clustering for text extraction and Tesseract and OmniPage optical character recognition (OCR) for recognizing text. An RGB-D camera-based navigation system is proposed in [20]. The camera motions are estimated by utilizing

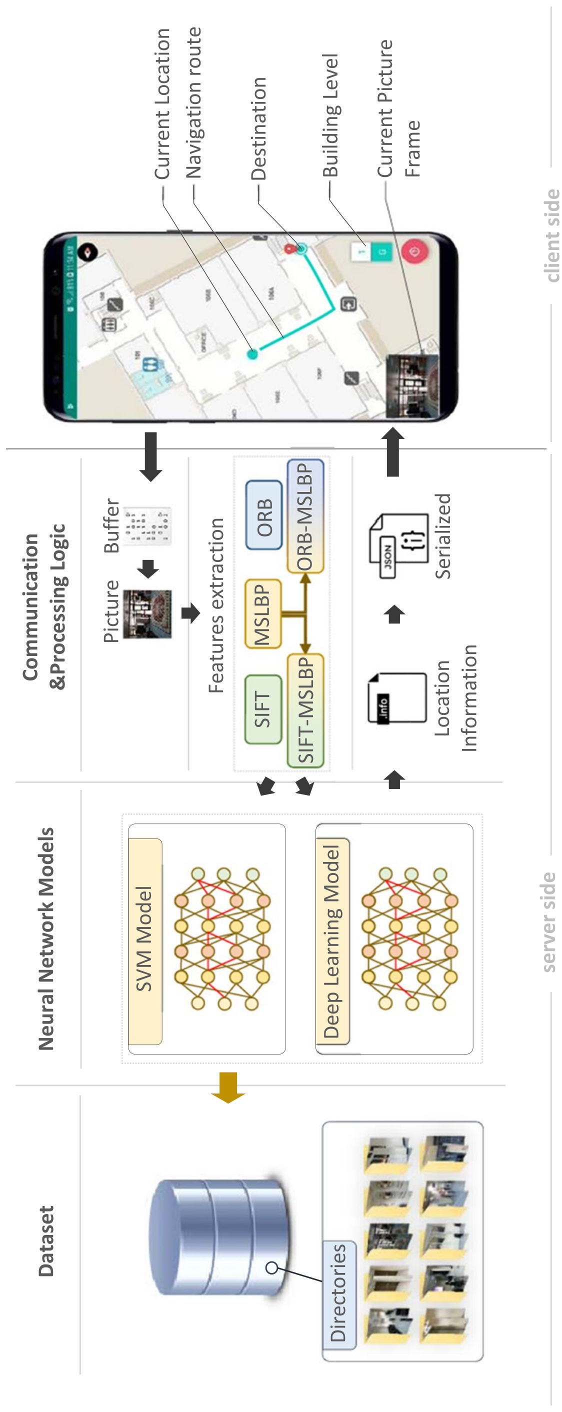

Fig. 1 System architecture. The server side is composed of three main components: (1) communication and processing Logic, (2) neural network models, and (3) a dataset. The client side is the mobile end-user application

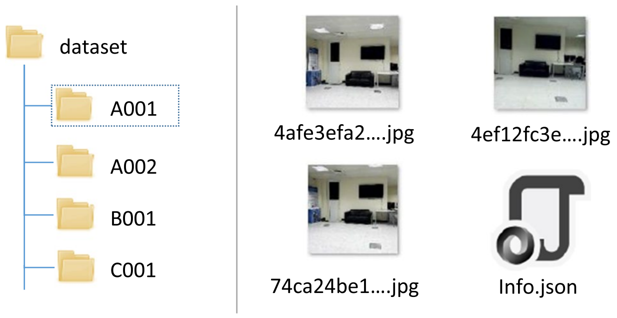

Fig. 2 Indoor dataset sample directories

sparse features and dense point clouds. A corner-based real-time motion estimator algorithm was adopted for *pose* estimation of the objects in the navigation path. The limitations of map merging result in inconsistency in the constructed map and, therefore, inaccuracy in the estimation. In addition, Google Glass [35] and Android smartphones were employed to aid people with VI to navigate indoor areas [12]. The proposed system utilized the Canny edge detector and Hough line transform for object detection tasks. A floor detection algorithm has also been employed to estimate the distance to walls. In [22], an indoor navigation system designed for people with VI has also been proposed. It uses the image subtraction method for detecting objects. A histogram backpropagation algorithm was employed to create color histograms of detected objects. The continuously adaptive mean shift algorithm was also used for tracking users.

3DLoc [13] is an indoor positing system that is based on the 3D features of indoor areas. 3D signatures of the different indoor places were recorded to recognize these places. A 3D signature extraction algorithm was proposed in 3DLoc. The proposed 3D algorithm uses indoor geometric reasoning for constructing 3D models from the captured images. The K-location algorithm was also introduced to identify locations. Kawaji et al. [18] proposed an indoor localization method that uses principal component analysis (PCA) with SIFT, that is, PCA-SIFT. Euclidean distance with locality-sensitive hashing was used to match the captured pictures with locations. In fact, PCA-SIFT increased the feature extraction time from SIFT and speeded-up robust features (SURF); however, the overall prediction time of the system was reduced. Xiao et al. [41] proposed a smartphone-based indoor positioning system that utilizes static objects such as doors and windows as references for positioning. The proposed system uses a deep learning model for recognizing static objects. The SIFT method for feature extraction was also utilized for position estimation. The obtained assessment results demonstrated that the system is capable of achieving an accuracy of 1 meter. Chen et al. [6] proposed a visual indoor positioning system that utilizes a convolutional neural network (CNN) for image retrieval. The trained deep learning VGG-16 network [32] was used to retrieve the most similar images with respect to query images. ORB features [30] of retrieved images and query images were utilized for the position estimation task. The average error in position estimation was less than 0.35 meters.

Adorno et al. [1] proposed a smartphone-based floor detection method to ensure the safety of elderly people while accessing indoor environments. The proposed system utilized a superpixel segmentation method to detect floors in unstructured indoor environments. The Hough line transform and corner detection were used for floor detection tasks in structured environments. The results show that the system is capable of recognizing with accuracies of 87.6% and 93% for unstructured environments and structured environments, respectively. Another indoor navigation system that uses GIST descriptors was proposed in [3]. Euclidean distance between descriptors was used for localization tasks. The employment of GIST descriptors enhanced the processing of the images and reduced the memory requirements. Murillo et al. [23] proposed an omnidirectional wearable system for providing guidance to the use while accessing indoor environments. The proposed system integrated SURF descriptors with GIST feature descriptors to create feature vectors of images. The visual odometry module in the system was composed of the extended Kalman filter-based SLAM and omnidirectional sensors.

Tian et al. [36] proposed a computer vision-based door detection method for assisting blind people in accessing unfamiliar indoor environments. A miniature camera mounted on the head via a cap or sunglasses captures the image, and a computer provides the processing and speech output. Door detection is based on a generic geometric door model built on stable edge and corner features. The demonstrated result indicates that the true positive rate is 91.9%. A computer vision module for assisting people with VI to access indoor and outdoor areas was proposed in [8]. An image optimization algorithm and a two-layer disparity image segmentation were used to detect the objects in indoor areas. To ensure their safety, the proposed method analyzes the depth of information at one to two meters. Cheng et al. [7] proposed a real-time indoor navigation system that is based on a feature-matching algorithm. The authors noted that traditional SIFT feature-based image matching is a time-consuming process. For this reason, the authors proposed their own technique to accelerate features matching. For this reason, they introduced GPU-accelerated SIFT for real-time image matching. The obtained assessment results demonstrated that GPU-accelerated SIFT has improved computational efficiency and reduced mismatching of features over other methods. However, the continuous employment of DL on mobile devices reduces the battery life dramatically [11]. A wearable virtual usher to aid people with VI in accessing indoor areas was proposed in [21]. The proposed system is composed of a wearable camera to extract scenes in front of the users. A headphone that is connected to a laptop is used to provide routing guidance. In the current design of the system, the users are required to carry the laptop on their back. The system uses egocentric visual perception to aid users. The SIFT algorithm is used for the indoor scene recognition process. A self-adaptive dynamic Bayesian network (SA-DBN) is used to find the best navigation route.

Zheng et al. [42] proposed an RGB-D camera-based indoor localization method in low light scenarios. RGB-D cameras are capable of providing normal RGB images with depth values for each pixel. The ORB feature detector algorithm was used to extract features from these images. However, not all mobile devices provide RGB-D images. iNavigation [39], an indoor navigation system, has also been proposed. It uses extracted SIFT features to match locations. An approximate nearest neighbor (ANN) search algorithm was used to match locations [4]. The ANN demonstrated its efficiency in providing higher accuracy than SVM. However, the SVM algorithm performs much better when a large number of pictures are used. Moreover, the time required to process images in SVM is much lower than the required time in ANN [27]. Travi-Navi [43], a vision-guided indoor navigation system, has also been proposed. It uses ORB features to match captured images with previously collected images. The constructed histograms from ORB features were employed over SVM to find the best classification. Wi-Fi fingerprints and magnetic fields were used to provide concrete systems. To concentrate only on the computer-vision side, we noted that the employment of MSLBP features can enhance the recognition rate or accuracy of matching images.

In summary, a considerable number of indoor navigation systems have been proposed. These systems use diverse kinds of features, such as SIFT [39], edge detection [5], ORB [43], and other methods. However, based on a recent study, SIFT and ORB demonstrated their competency when employed for recognition in complex scenarios [17]. But a system may fail in recognizing locations while using pictures that do not have extractable ORB features. For this reason, we consider the usage of SIFT, ORB, MSLBP and the combination of SIFT and ORB with MSLBP features to improve the performance.

3 Proposed system

In this section, we present the different components of CamNav. We first detail our approach based on multiscale local binary patterns (MSLBP) [34]. Therefore, we present a combination of ORB and SIFT with MSLBP features. Eventually, in this section, we present our developed DL model that is used to predict locations using extracted features.

3.1 Multiscale locale binary pattern features

Multiscale local binary pattern (MSLBP) features have been considered for recognizing places. MSLBP has been compared with systems that use ORB features [43] and SIFT features [39]. Local binary pattern (LBP) is an effective texture classification technique that annotates the image's pixels through the thresholding neighborhood of each pixel, resulting in a binary number [25]. The main benefit of using a local binary pattern is its computational simplicity, which enables processing the image in a challenging, real-time fashion. The notion of multiscale in LBP is to consider more or fewer neighbors when computing the binary value of a pixel [26]. For instance, when using the scale of eight (S=8) of an image I with radius equal to eight (r=8), there will be a total of 256 patterns, out of which 58 are uniform, which gives 59 labels [25]. In our work, we divide the pictures into 4 parts, and we consider, for each part, eight scales from 1 to eight, as shown in the algorithm in Algorithm 1. The algorithm takes the path of the dataset P_{DS} as input and passes over the directories, or classes, inside it. For each directory, all images will be retrieved using the *getImages*(*path*) function. We retrieve each image separately, and we obtain the LBP features for the four parts (top-left, top-right, bottom-left, and bottom-right parts) using the function *lbp*. We put the LBP features together in a vector including the obtained LBP features of all scales. We create a hash table and store the computed LBP features next to the unique name of the image. The extracted features are therefore clustered to form a dictionary. A histogram is constructed for each picture to measure the frequency of every word in the dictionary as in [43]. The histograms of all the pictures show a fixed-size array of length 100. The SVM model is then trained using the histograms of pictures.

Algorithm 1: Extracting multiscale local binary pattern features

```
Input: DataSet Path $P_{DS}$
Initialization: $ht_{mslbp} \leftarrow$ new $HashTable()$
for each $string\ directory$ in $getDirectories(P_{DS})$ do
    for each $imagePath$ in $getImages(directory)$ do
        I $\leftarrow imread(imagePath)$                ▷ reading the image
        $r \leftarrow 8$                                 ▷ radius $r$ is 8
        $LBP_I \leftarrow [\,]$                          ▷ empty matrix row
        for $scale \leftarrow 1$ to $8$ do
            $i_1 \leftarrow$ I(1:112, 1:112)             ▷ top-left part
            $f_1 \leftarrow$ lbp($I_1$,$scale$,$r$)
            $i_2 \leftarrow$ I(1:112, 113:end)           ▷ top-right part
            $f_2 \leftarrow$ lbp($I_2$,$scale$,$r$)
            $i_3 \leftarrow$ I(113:end, 1:112)           ▷ bottom-left part
            $f_3 \leftarrow$ lbp($I_3$,$scale$,$r$)
            $i_4 \leftarrow$ I(113:end, 113:end)         ▷ bottom-right part
            $f_4 \leftarrow$ lbp($I_4$,$scale$,$r$)
            $LBP_I \leftarrow [LBP_I\ f_1\ f_2\ f_3\ f_4]$
        end
        $ht_{mslbp}[imagePath] \leftarrow LBP_I$
    end
end
Output: $ht_{mslbp}$
```

3.2 Combination of ORB and SIFT with MSLBP

When building the histogram of ORB and SIFT features, we consider combining them with the extracted MSLBP features. We use the notion of ORB-MSLBP and SIFT-MSLBP for the combined histograms with MSLBP features. Therefore, the combined histograms are used individually to train the SVM model and to analyze the combination effects on the prediction results. In fact, the number of extracted ORB and SIFT features differs from one picture to another. The shape S of the ORB and SIFT matrix is $S_{\text{orb}} = 32 \times n$ and $S_{\text{SIFT}} = 128 \times n$, respectively, where n is an unknown number of key features (rows). In contrast, the shape of the extracted MSLBP features in our case is always constant, having a single row matrix of size 1024. For this reason, to merge it with ORB, the row of MSLBP is reshaped to 32 elements in width resulting in a 32×32 matrix, where 32 is the number of rows. Therefore, the reshaped MSLBP matrix is appended to the matrix of ORB features to form ORB-MSLBP. We perform the same procedure for SIFT-MSLBP by reshaping MSLBP features into 128 elements in width,

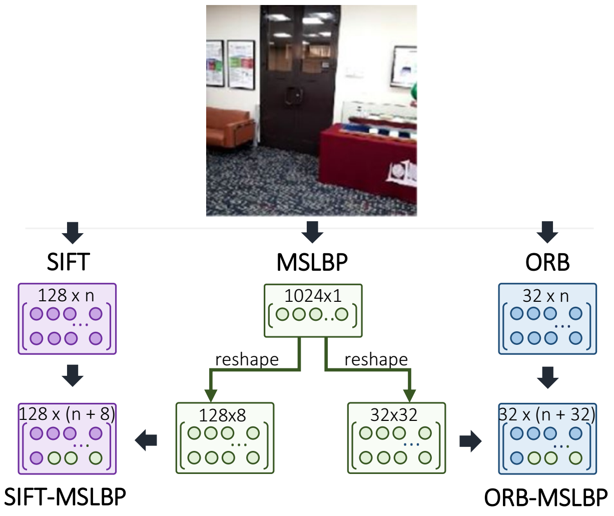

Fig. 3 Combining SIFT features with MSLBP features

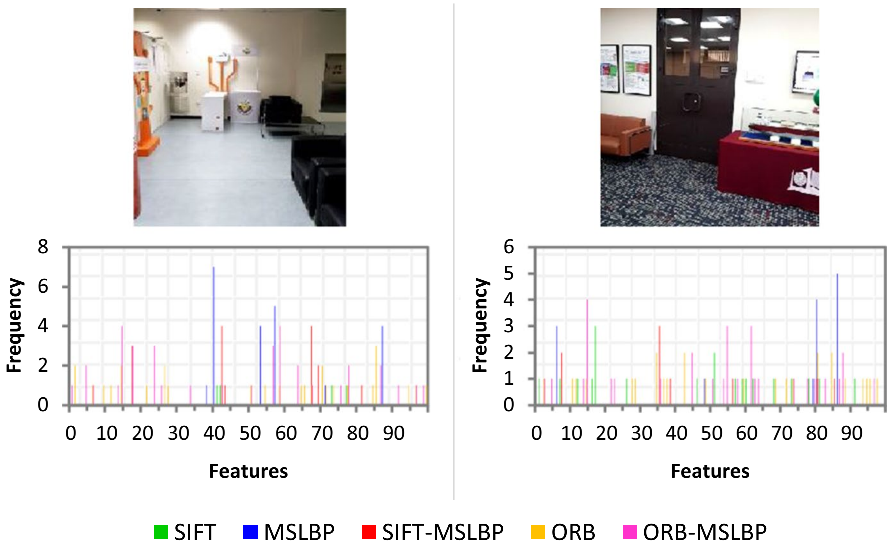

Fig. 4 Histograms of the different types of features for two pictures

resulting in a 128×8 matrix, where 8 is the number of rows. Figure 3 depicts the flow of reshaping and appending MSLBP features. Consequently, the histograms of the new, combined matrices are constructed as discussed in the previous section. Figure 4 illustrates the histogram of two different pictures showing all ORB, SIFT, MSLBP and the combined ORB-MSLBP and SIFT-MSLBP features.

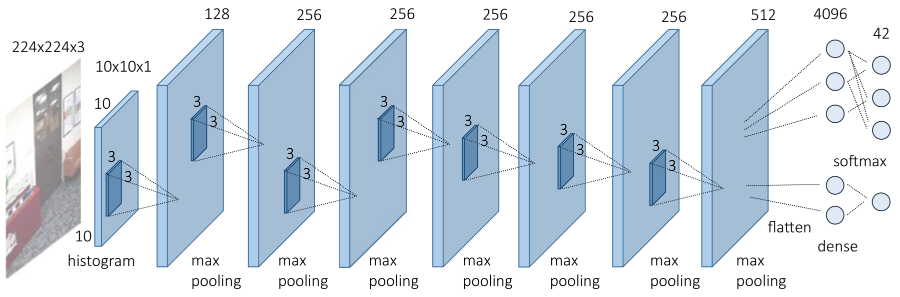

Fig. 5 The architecture of the developed deep learning model

3.3 Indoor deep learning model

We built a deep learning (DL) model to recognize places using ORB, SIFT, MSLBP, and ORB-MSLBP and SIFT-MSLBP histograms. The constructed dataset is used to extract the required features and to train three models individually for each type. The architecture of the model is illustrated in Fig. 5. We do not compare the accuracy with different DL models to focus only on the analysis of ORB, SIFT and MSLBP features. We used TensorFlow[1], an open-source deep learning framework, to create the model. The model is composed of 16 layers. The input of the model is an array of size 100 shaped to a 10×10 matrix. The output of the model is the classification index of the entered features. During the training phase, we keep the training process running until the model reaches the optimal stabilized accuracy and precision. In other words, the training stops immediately after the number of n iterations obtains the highest accuracy and precision. Each model is currently trained with the features that are extracted from 4,200 images to identify 42 different places. In addition, we consider training the DL model separately with the flat features of MSLBP without using the histograms of features. For this reason, we changed the input shape of the DL model from 10×10 to a 32×32 matrix to match the size of the MSLBP array, which is an array composed of 1024 numbers.

4 System development and technical details

The architecture of our system is a client–server architecture. The server part is responsible for performing complex processing computations. In fact, we have noted that the use of image processing, as well as deep learning techniques, on a mobile device consumes a considerable amount of processing resources, resulting in a significant loss in battery life. The client side is the mobile application that provides the services of indoor positioning and navigation. The mobile application is configured to send captured images to the server in real time and wait for their recognitions. To

[1] TensorFlow: https://www.tensorflow.org/.

compare the power consumption on the mobile device, we configured the mobile application to perform processing on itself and on the server. Figure 5 shows the complete architecture of the developed model. We detail the constructed dataset, the server application, and the end-user mobile application in the following sections.

4.1 Indoor map and dataset

To present the map, we contacted the Design and Architecture Center at Qatar University to provide us with computer-aided drawing (.cad) files of one building. We uploaded them into MapBox to make them accessible from the mobile device. We marked the desired locations with a unique location identifier (ID), and we collected a total of 5,208 pictures to build our indoor dataset[2]. The dataset is used to obtain required information for relevant locations and to train both SVM and the developed deep learning model. The dataset is composed of a set of directories in which each represents one indoor location. Each directory represents one class. Each class contains 100 pictures along with one information file (info.json) and 24 pictures dedicated for testing purposes. To account for the different orientations of users, we collected pictures from different angles for the same places. Pictures were taken from diverse kinds of mobile devices, including the Samsung Galaxy S8 and the LG Nexus 5. We considered the diversity of mobile phones to account for the different types of pictures that are taken from a variety of cameras. The size of the images stored in the dataset is 224 in height and width. We manually annotated the detailed data in the information files at the current stage. Information files include latitude, longitude, level, the building ID, and the location ID. The information file is loaded when a captured image is processed.

4.2 Server application

We have placed the modules that require tremendous processing in a separate application on a server device. This includes the logic of processing the pictures, training the SVM and deep learning models, loading the models, and predicting the locations from the images. We configured the server application with ActiveMQ, an open source message broker for real-time messaging services, to receive captured pictures and send back the processed results. The server device requires NVIDIA[3] graphical processing unit GPU Graphic card with CUDA compute capability of 3.5 or more. This requirement enables the execution of TensorFlow in GPU mode.

4.3 Mobile application

We have implemented a mobile user application and configured it to capture pictures in real time. The mobile application is configured to send, by default, the buffer of

[2] CamNav-dataset: https://github.com/akarkar/CamNav-dataset.

[3] NVIDIA Technology: https://www.nvidia.com/ .

captured images to the server to process them. The server returns the processed results, in JSON format, to the mobile application. Therefore, the mobile device updates the location of the user on the map.

5 Evaluation

The proposed system, CamNav, is developed and compared with the usage of a SVM-based indoor navigation system [43]. We used in our study a laptop supplied with NVIDIA GTX 1060, 24 GB RAM and i7-6700HQ CPU (2.60 GHz). We developed several functionalities to collect data about the training accuracy, testing accuracy, CPU and memory usage, and the distribution of extracted features. We also performed several statistical analyses to provide valuable evaluations, such as analysis of variance (ANOVA), analysis of covariance (ANCOVA), and model selection. We used RStudio[4] to perform the required computations and the necessary statistical analyses. *P* values and *F*-values are used during the analysis. The *P* value represents the probability of finding statistical model at least as extreme as the observed one, assuming that the null hypothesis is true. Generally, for a *P* value that is less than or equal to 0.05, that is, the level of 5%, we reject the null hypothesis. *F*-statistics can be used to confirm the significance of the test. The *F*-value is the obtained ratio of mean-squares, which is equal to:

$$F_{\text{value}} = \frac{(\text{Differences between groups})}{(\text{Differences within groups})} \tag{1}$$

If the *F*-value surpasses the critical value, the null hypothesis will be rejected, and it is feasible to conclude that there is a significant effect in the assessment. In brief, the *P* value characterizes the probability, and the *F*-value is the obtained value of the test. The conducted assessment is based on the collected images, a total number of 5184 images, with height and width equal to 224 pixels each, and the trained models, including a support vector machine (SVM) and a deep learning (DL) model. The images were collected from one building located at Qatar University, namely B09 – Research Building. There were 42 classes in which each class had 100 images to be used for training and 24 images to be used for testing. We noted that some pictures do not have extractable ORB features, so we eliminated them from the training set. However, pictures that do not have extractable ORB or SIFT features are marked as *not recognized correctly*. We do not consider the training phase of the models in the performance analysis. The evaluation is categorized into three aspects: (a) the prediction accuracy, (b) the *number of features* as the dependent variable (DV), and (c) the prediction time.

[4] RStudio: http://www.rstudio.com/.

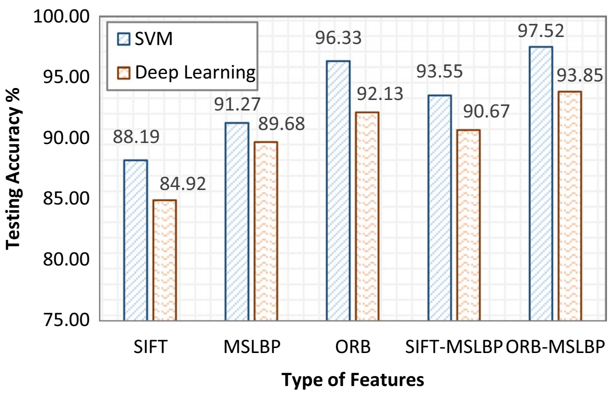

Fig. 6 Overall accuracy of the different feature types in SVM and deep learning

5.1 Prediction accuracy

To analyze the effect of using MSLBP features, we compared it to using ORB features that were used to match places in Travi-Navi [43] and SIFT in iNavigation [39]. In addition, ORB and SIFT were combined with MSLBP features. The histograms of the extracted features were used with a trained deep learning model. Figure 6 shows the results of the obtained results using the SVM and deep learning models. During the training phase, we noted that 54 images did not have ORB features. Since these pictures have MSLBP features, we do not consider these pictures while training the DL model when testing with ORB. However, we include all the pictures when testing with SIFT and MSLBP. In fact, the usage of MSLBP improved the prediction accuracy of identifying locations from 88.19% (using SIFT) to 91.27%. In addition, the combination of SIFT and MSLBP improved the prediction accuracy to 93.55%. In addition, the combination of ORB and MSLBP improved the prediction accuracy from to 96.33% to 97.52%. From the obtained histograms in the deep learning model, it can be seen that the SVM performs better in terms of predicting locations. This result demonstrates that deep learning is not performing better when it is applied to a small set of data.

We also predict the locations using the extracted MSLBP features in both SVM and in the deep learning model without building their histograms. When using the histograms, the obtained accuracies in SVM and deep learning are 97.53% and 93.85%, respectively. However, when using MSLBP flat features, with a sequence of 1024 numbers for a picture, the obtained accuracy results in SVM and the deep learning model are 4.33% and 99.21%, respectively. These results demonstrate that deep learning is capable of providing very high accuracy when using extracted MSLBP flat features that contain a sequence of 1024 numbers for each picture instead of using the histograms of features that contain a sequence of 100 numbers for each picture. Figure 7 illustrates the obtained results while using the histogram of SIFT-MSLBP features and the MSLBP flat features in both the SVM and deep learning models. The histogram of SIFT-MSLBP is considered instead of the

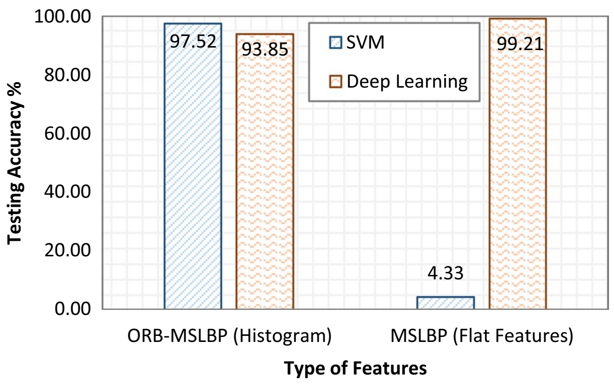

Fig. 7 Overall testing accuracy for images using ORB-MSLBP and MSLBP features while using SVM and deep learning

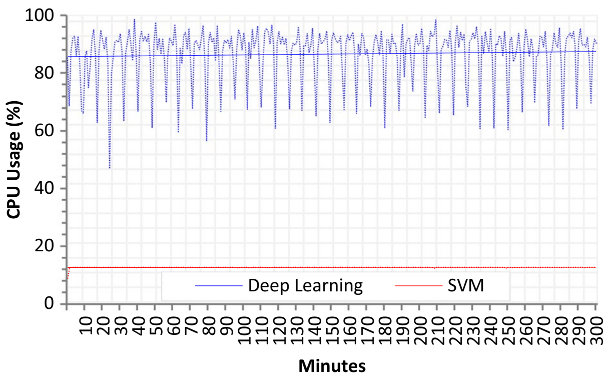

Fig. 8 CPU trace over 5 min when predicting using SVM and deep learning

histogram of MSLBP because SIFT-MSLBP gave better results. The reason for not using the flat features of SIFT in deep learning is that the number of extracted features differs from one image to another, which is unlike the case of MSLBP, where the number of extracted features is fixed (Fig. 8).

While the usage of the flat features of MSLBP with deep learning gives a very high accuracy, the CPU usage in deep learning is also very high compared with the CPU usage when using SVM, as shown in Fig. 8. In fact, the average CPU usage by the DL and SVM methods are 12 ± 0.22% and 87 ± 9.46%, respectively. This result demonstrates that the DL method consumes much more processing resources compared with the SVM method.

5.2 "Number of features" as a dependent variable

Figure 9 shows the number of extracted ORB and SIFT features of every image in each class. Each image can have a different number of extracted features. We

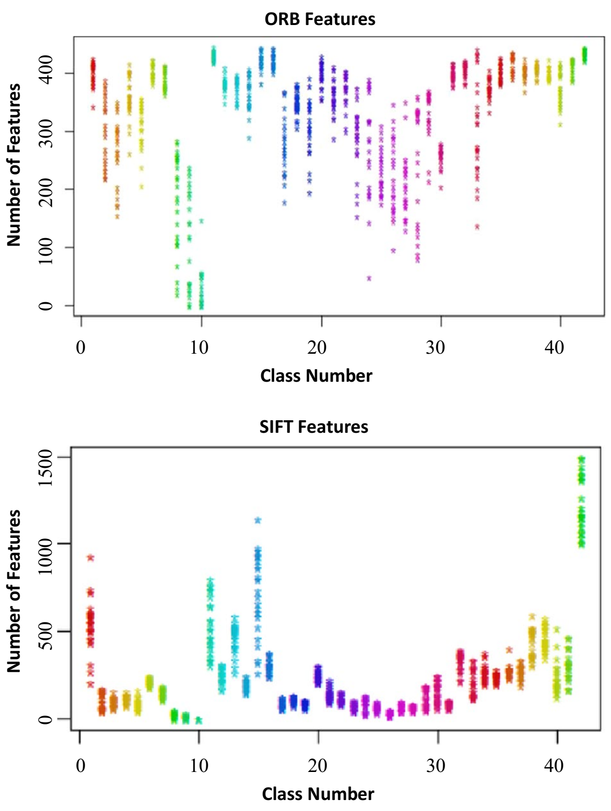

Fig. 9 Number of extracted ORB and SIFT features per image with 42 classes

study the effect of the *number of features* on the prediction result of images. To validate this assumption, the analysis of covariance (ANCOVA) is considered to assess the mean of the dependent variable (DV).

The linear model for the obtained predictions of the ORB, SVM, and DL models is considered for statistical treatment with the extracted *number of features* variable. The linear model for obtaining correct predictions according to the number of SIFT features is defined as follows:

$$f(x) = s_t x + c_t \tag{2}$$

where s is the slope, t is the neural network type, x is the number of extracted features, and c is the coefficient. Based on the obtained testing results, the coefficient and slope of the linear model for ORB are equal to:

$$c_{\text{orb}} = 0.685, s_{\text{orb}} = 7.83E^{-4}$$

respectively. The coefficient and slope of the linear model for SIFT are equal to:

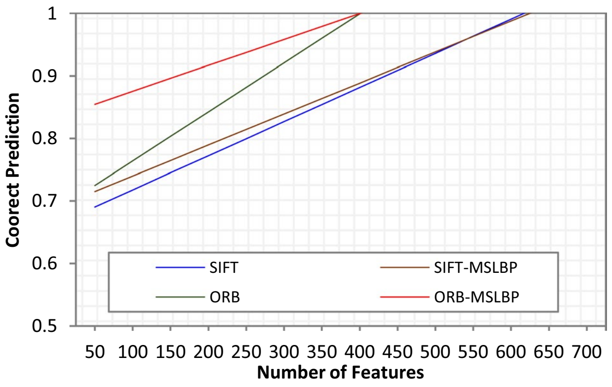

Fig. 10 Linear model for correct prediction according to the number of extracted features for ORB, SIFT, ORB-MSLBP, and SIFT-MSLBP

$$c_{\text{sift}} = 0.663, s_{\text{sift}} = 5.46E^{-4}$$

respectively. The coefficient and slope of the linear model for ORB-MSLBP are equal to:

$$c_{\text{orb-mslbp}} = 0.663, s_{\text{orb-mslbp}} = 4.14E^{-4}$$

respectively, and the coefficient and slope of the linear model for SIFT-MSLBP are equal to:

$$c_{\text{sift-mslbp}} = 0.690, s_{\text{sift-mslbp}} = 4.95E^{-4}$$

respectively. Figure 10 shows the graph of the linear models, taking into consideration the correct prediction and the number of extracted ORB and SIFT features. In fact, the linear models are heading upward, which implies that the probability of achieving higher prediction while recognizing the correct location of images can be achieved by images that have a higher number of features. Since the slope of f_{orb} is larger than the slope of $f_{\text{orb-mslbp}}$ ($s_{\text{orb}} > f_{\text{orb-mslbp}}$), this means that prediction using ORB will provide better prediction with a larger number of extracted features, while prediction with ORB-MSLBP can provide better prediction even with a lower number of extracted ORB features. Thus, obtaining the correct prediction with MSLBP is less dependent on the number of ORB features; the coefficient of ORB-MSLBP is larger than the coefficient of ORB ($c_{\text{orb}} > c_{\text{orb-mslbp}}$). The same result occurs for SIFT and SIFT-MSLBP. We can also conclude that the use of MSLBP reduces the dependency on the number of features since both slopes $s_{\text{orb-mslbp}}$ and $s_{\text{sift-mslbp}}$ are smaller than s_{orb} and s_{sift}, respectively ($s_{\text{orb-mslbp}} < s_{\text{orb}}$ and $s_{\text{sift-mslbp}} < s_{\text{sift}}$).

5.3 Prediction time

A script was written to track the prediction time when using the different types of features, SIFT, MSLBP, ORB, SIFT-MSLBP, and ORB-MSLBP. Figure 11

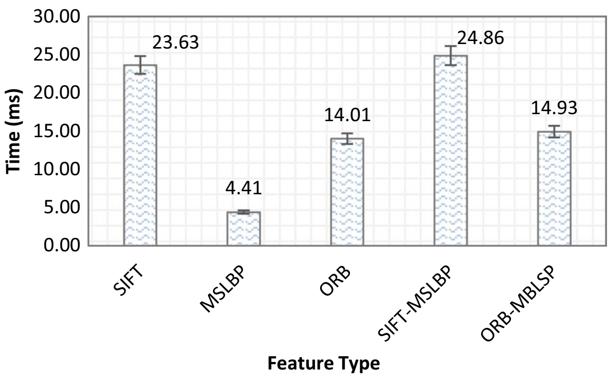

Fig. 11 Time consumed while classifying 1,008 pictures for location identification using the different features, SIFT, MSLBP and SIFT-MSLBP in SVM

illustrates the consumed time when predicting locations. The usage of MSLBP significantly improved the prediction time by decreasing it to 4.41 ± 0.64 ms from 14.01 ± 2.47 ms from ORB. The prediction time of ORB-MSLBP, which is 14.93 ± 2.06 ms, is slightly larger than the prediction time used by ORB, because more features are being considered when using the histogram. Based on the time consumed, it can be seen that predicting the location of pictures while using MSLBP features is more powerful than using ORB features alone. However, the little increase in accuracy will affect the recognition of places and thus guiding the users, especially people with visual impairments. For this reason, when the higher prediction accuracy is needed, the integration of ORB-MSLBP is recommended.

5.4 Mobile power consumption

We studied the power consumption when processing pictures on the mobile device and while processing them on the server side. We ran the application in 4 modes: (a) *idle mode*: the application is only capturing pictures and not performing any processing; (b) *SVM mode*: the application is processing captured pictures using SVM model on the phone; (c) *DL mode*: the application is processing the pictures using deep learning mode on the phone; and (d) *on-server mode*: the application is processing the pictures on the server side, but it only reshapes pictures on the mobile device, sends the buffer to the server taking in consideration splitting it to chunks if required, and displays the result. A Galaxy S8+ phone was used for each mode for a period of 20 minutes to measure the energy consumption of the battery. The application is configured to capture pictures continuously for every period of at least 500 ms. Forecasting [40] the level of the battery is also considered after tracing the battery level for 20 minutes. Figure 12 illustrates the energy consumption of the battery while using different modes of application.

Based on the obtained results, performing the processing on the server will save mobile device power. If the application is running in idle mode, it consumes the minimum level of the battery; thus, the battery will last for approximately 292 minutes.

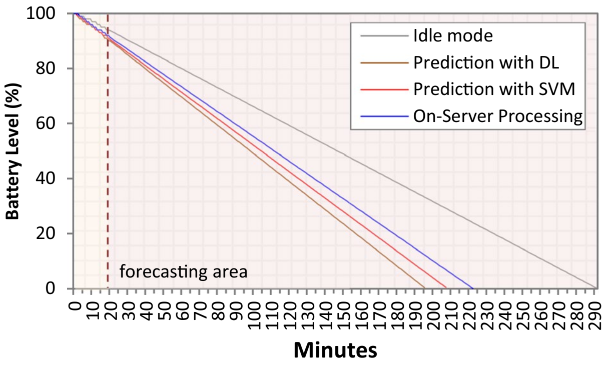

Fig. 12 Change in battery level over time for the different modes

The use of DL for prediction on the phone will cause the highest battery consumption, and the battery will become empty after approximately 196 minutes. The prediction using SVM will consume the battery after approximately 208 minutes, which is better than DL. Eventually, the on-server prediction will keep the mobile device alive for a longer time: 223 minutes. Based on this study, it can be seen that processing resources on the server will save more power on mobile devices. While processing on server requires the involvement of that wireless adapter, the intensive use of wireless communication remains better than the intensive use of processing on the mobile device itself. Since our research work focuses mainly on the improvement of recognizing places using computer vision techniques, therefore further assessment on the power consumed by the wireless adapter, central processing unit (CPU) and screen characteristics with diverse sort of mobile devices can be considered in future studies.

6 Conclusion

In this paper, we presented CamNav, an indoor navigation mobile-based system that uses MSLBP features to recognize places. The system has been compared with existing systems that use ORB and SIFT features during the recognition phase of places. The usage of MSLBP features improved the overall place-recognition accuracy result from 96.33% (from ORB) to 97.02%. In addition, the combination of SIFT features and MSLBP features improved the accuracy from 88.00% to 94.15%. A deep learning model was also constructed to predict images with the flat form of MSLBP features. Battery consumption of mobile devices was also studied to demonstrate that processing data on the server will save more power. Thus, while processing on the server, the battery will last for approximately 223 minutes instead of 208 minutes. Our system is not capable at the current stage to differentiate between different locations that look similar. Future work will consider the precise localization of users through the use of simultaneous localization and mapping. Potential solutions for pictures with no available ORB or SIFT features will be examined.

Enhanced power consumption will be studied more. Usability analyses by both healthy people and people with visual impairments are also planned.

Acknowledgements This publication was supported by Qatar University Collaborative High Impact Grant QUHI-CENG-18/19-1. The findings achieved herein are solely the responsibility of the authors. The contents of this publication are solely the responsibility of the authors and do not necessarily represent the official views of the Qatar University.

References

1. Adorno J, DeLaHoz Y, Labrador MA (2016) Smartphone-based floor detection in unstructured and structured environments. In: 2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). IEEE, pp 1–6
2. Ahmetovic D, Gleason C, Ruan C, Kitani K, Takagi H, Asakawa C (2016) Navcog: a navigational cognitive assistant for the blind. In: Proceedings of the 18th International Conference on Human–Computer Interaction with Mobile Devices and Services. ACM, pp 90–99
3. Athira SV, George M, Jose BR, Mathew J (2017) A global image descriptor based navigation system for indoor environment. Procedia Comput Sci 115:466–473
4. Beis JS, Lowe DG (1997) Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In: CVPR, vol 97. Citeseer, p 1000
5. Canny J (1987) A computational approach to edge detection. In: Readings in Computer Vision. Elsevier, Amsterdam, pp 184–203
6. Chen Y, Chen R, Liu M, Xiao A, Wu D, Zhao S (2018) Indoor visual positioning aided by CNN-based image retrieval: training-free, 3d modeling-free. Sensors 18(8):2692
7. Cheng J, Zhu X, Ding W, Gao G (2016) A robust real-time indoor navigation technique based on GPU-accelerated feature matching. In: 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN). IEEE, pp 1–4
8. Costa P, Fernandes H, Martins P, Barroso J, Hadjileontiadis LJ (2012) Obstacle detection using stereo imaging to assist the navigation of visually impaired people. Procedia Comput Sci 14:83–93
9. Deniz O, Paton J, Salido J, Bueno G, Ramanan J (2014) A vision-based localization algorithm for an indoor navigation app. In: 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies. IEEE, pp 7–12
10. Doush IA, Alshatnawi S, Al-Tamimi AK, Alhasan B, Hamasha S (2017) Isab: integrated indoor navigation system for the blind. Interact Comput 29(2):181–202
11. Eshratifar AE, Pedram M (2018) Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment. In: Proceedings of the 2018 on Great Lakes Symposium on VLSI. ACM, pp 111–116
12. Garcia G, Nahapetian A (2015) Wearable computing for image-based indoor navigation of the visually impaired. In: Proceedings of the Conference on Wireless Health. ACM, p 17
13. Huang Z, Gu N, Hao J, Shen J (2018) 3dloc: 3d features for accurate indoor positioning. Proc ACM Interact Mob Wear Ubiq Technol 1(4):141
14. Ivanov R (2012) Rsnavi: an RFID-based context-aware indoor navigation system for the blind. In: Proceedings of the 13th International Conference on Computer Systems and Technologies. ACM, pp 313–320
15. Jafri R, Campos RL, Ali SA, Arabnia HR (2018) Visual and infrared sensor data-based obstacle detection for the visually impaired using the Google project tango tablet development kit and the unity engine. IEEE Access 6:443–454
16. Kacorri H, Mascetti S, Gerino A, Ahmetovic D, Takagi H, Asakawa C (2016) Supporting orientation of people with visual impairment: analysis of large scale usage data. In: Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility. ACM, pp 151–159
17. Karami E, Prasad S, Shehata M (2017) Image matching using sift, surf, brief and orb: performance comparison for distorted images. arXiv preprint arXiv:1710.02726
18. Kawaji H, Hatada K, Yamasaki T, Aizawa K (2010) Image-based indoor positioning system: fast image matching using omnidirectional panoramic images. In: Proceedings of the 1st ACM International Workshop on Multimodal Pervasive Video Analysis. ACM, pp 1–4

19. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
20. Lee YH, Medioni G (2016) RGB-D camera based wearable navigation system for the visually impaired. Comput Vis Image Underst 149:3–20
21. Li L, Xu Q, Chandrasekhar V, Lim JH, Tan C, Mukawa MA (2016) A wearable virtual usher for vision-based cognitive indoor navigation. IEEE Trans Cybern 47(4):841–854
22. Manlises C, Yumang A, Marcelo M, Adriano A, Reyes J (2016) Indoor navigation system based on computer vision using camshift and d* algorithm for visually impaired. In: 2016 6th IEEE International Conference on Control System, Computing and Engineering (ICCSCE). IEEE, pp 481–484
23. Murillo AC, Gutiérrez-Gómez D, Rituerto A, Puig L, Guerrero JJ (2012) Wearable omnidirectional vision system for personal localization and guidance. In: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. IEEE, pp 8–14
24. Noh Y, Yamaguchi H, Lee U (2018) Infrastructure-free collaborative indoor positioning scheme for time-critical team operations. IEEE Trans Syst Man Cybern Syst 48(3):418–432
25. Pietikäinen M (2010) Local binary patterns. Scholarpedia 5(3):9775
26. Pietikäinen M, Hadid A, Zhao G, Ahonen T (2011) Computer vision using local binary patterns, vol 40. Springer, Berlin
27. Raikwal J, Saxena K (2012) Performance evaluation of SVM and k-nearest neighbor algorithm over medical data set. Int J Comput Appl 50(14):35–39
28. Raja Y, Gong S (2006) Sparse multiscale local binary patterns. In: BMVC, pp 799–808
29. Riggs W, Gordon K (2017) How is mobile technology changing city planning? developing a taxonomy for the future. Environ Plan B Urban Anal City Sci 44(1):100–119
30. Rublee E, Rabaud V, Konolige K, Bradski GR (2011) Orb: an efficient alternative to sift or surf. In: ICCV, vol 11. Citeseer, p 2
31. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
32. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
33. Sonka M, Hlavac V, Boyle R (2014) Image processing, analysis, and machine vision. In: Cengage Learning
34. Srivastava P, Khare A (2018) Utilizing multiscale local binary pattern for content-based image retrieval. Multimed Tools Appl 77(10):12377–12403
35. Starner T (2013) Project glass: an extension of the self. IEEE Pervas Comput 12(2):14–16
36. Tian Y, Yang X, Arditi A (2010) Computer vision-based door detection for accessibility of unfamiliar environments to blind persons. In: International Conference on Computers for Handicapped Persons. Springer, Berlin, pp 263–270
37. Tian Y, Yang X, Yi C, Arditi A (2013) Toward a computer vision-based wayfinding aid for blind persons to access unfamiliar indoor environments. Mach Vis Appl 24(3):521–535
38. Tuta J, Juric MB (2018) Mfam: multiple frequency adaptive model-based indoor localization method. Sensors 18(4):963
39. Wang E, Yan W (2014) inavigation: an image based indoor navigation system. Multimed Tools Appl 73(3):1597–1615
40. Weigend AS (2018) Time series prediction: forecasting the future and understanding the past. Routledge, London
41. Xiao A, Chen R, Li D, Chen Y, Wu D (2018) An indoor positioning system based on static objects in large indoor scenes by using smartphone cameras. Sensors 18(7):2229
42. Zheng Y, Luo P, Chen S, Hao J, Cheng H (2017) Visual search based indoor localization in low light via RGB-D camera. World Acad Sci Eng Technol Int J Comput Electr Autom Control Inf Eng 11(3):349–352
43. Zheng Y, Shen G, Li L, Zhao C, Li M, Zhao F (2017) Travi-navi: self-deployable indoor navigation system. IEEE/ACM Trans Netw 25(5):2655–2669