Electronics and Electrical Engineering

http://www.aimspress.com/journal/ElectronEng/

AIMS Electronics and Electrical Engineering, 1(1): 74–99
DOI: 10.3934/ElectrEng.2017.1.74
Received: 15 August 2017
Accepted: 28 November 2017
Published: 13 December 2017

Research article

Real-time indoor assistive localization with mobile omnidirectional vision and cloud GPU acceleration

Feng Hu [1,2,*], **Zhigang Zhu** [1,2], **Jeury Mejia** [3], **Hao Tang** [4], **and Jianting Zhang** [1,2]

[1] Department of Computer Science, The Graduate Center, City University of New York, New York City, NY, United States
[2] Department of Computer Science, The City College, City University of New York, New York City, NY, United States
[3] Department of Computer Science, Rutgers University, New Brunswick, NJ, United States
[4] Department of Computer Information Systems, Borough of Manhattan Community College, City University of New York, New York City, NY, United States

* **Correspondence:** Email:fhu@gradcenter.cuny.edu; Tel:+1-212-650-8799; Fax:+1-212-650-6248.

Abstract: In this paper we propose a real-time assistive localization approach to help blind and visually impaired people in navigating an indoor environment. The system consists of a mobile vision front end with a portable panoramic lens mounted on a smart phone, and a remote image feature-based database of the scene on a GPU-enabled server. Compact and effective omnidirectional image features are extracted and represented in the smart phone front end, and then transmitted to the server in the cloud. These features of a short video clip are used to search the database of the indoor environment via image-based indexing to find the location of the current view within the database, which is associated with floor plans of the environment. A median-filter-based multi-frame aggregation strategy is used for single path modeling, and a 2D multi-frame aggregation strategy based on the candidates' distribution densities is used for multi-path environmental modeling to provide a final location estimation. To deal with the high computational cost in searching a large database for a realistic navigation application, data parallelism and task parallelism properties are identified in the database indexing process, and computation is accelerated by using multi-core CPUs and GPUs. User-friendly HCI particularly for the visually impaired is designed and implemented on an iPhone, which also supports system configurations and scene modeling for new environments. Experiments on a database of an eight-floor building are carried out to demonstrate the capacity of the proposed system, with real-time response (14 fps) and robust localization results.

Keywords: assistive indoor localization; real-time system; GPU acceleration; mobile computing; omnidirectional vision

1. Introduction

An indoor localization system is of significant importance to the visually impaired in their daily lives by helping them localize themselves and further navigate unfamiliar indoor environments. There are 285 million visually impaired people in the world according to the World Health Organization (WHO) [*], among who 39 million are blind. Despite a large amount of research have been carried out for robot navigation in the robotics community [1–3], and several assistive systems are designed for blind people [4–8], efficient and effective portable solutions for visually impaired people are not yet available. In this paper, we intend to build an accurate and robust localization and navigation system for visually-impaired people, many of whom have adopted smart phones for their daily uses. For dealing with the difficulty of a visually impaired person aiming the camera, we mount a portable omnidirectional lens on the smart phone camera to construct a real-time imaging system with a 360-degree horizontal field of view. A client/server architecture is also used to ease the computational burden of the smart phone for real-time performance and the storage requirement of a large image feature-based database for a realistic indoor environment.

Currently, the mainstream solutions of assistive localization for the blind are based on GPS signals; however, in an indoor environment these methods are not applicable since GPS signals are unavailable or inaccurate. Pose measurements using other on-board sensors such as gyroscopes, pedometers, or IMUs, are not precise enough to provide user heading information and instructions for moving around for a visually impaired person.

There are some other alternative sensors besides GPS for providing localization services, such as RFID or bluetooth sensors. Kulyukin et al. [9] and Cicirelli et al. [10] develop an indoor localization system using RFID sensors. Although these RFID tags can passively integrate local navigation measurements to achieve global navigation objectives, the system heavily relies on the dense distribution of RFID sensors and specially designed robots. Some of them such as [9] also need to build specialized robots for each visually impaired person, which may cost thousands of dollars. Regardless, the RFID and bluetooth based approaches need to install vast amount of RFID sensors (or bluetooth beacons). The larger the areas need to be covered and the higher the accuracy is required, the more sensors are needed—leading theses solutions to be more expensive. In our method, no extra large amount of sensors or infrastructure (such as beacons, or RFID tags) need to be installed in the environment, and no other complex devices are required except a daily-used smart phone (such as an iPhone, which many blind people use daily) as well as a low-cost compact lens (usually such a lens is less than 100 dollars, including a smart phone case). Another existing solution proposed by Legge et al. [11] uses a handheld sign reader and widely distributed digitally encoded signs to give location information to the visually impaired. Again, this method requires attaching new tags to the environment, and it can only recognize some specific locations. Our system does not have requirements for changing the environment, and the viewer can be localized in the entire interiors of a building instead of just a few individual locations.

One example of our testing environments is shown in Figure 1. The map in the middle is an indoor floor plan of one floor of a campus building. The red line in the map is one of the traversal paths when modeling this environment. The blue and green lines are the paths used for capturing test video clips. Some sample omnidirectional images are shown around the map, and their geo-locations are indicated in the floor plan.

[*] http://www.who.int/mediacentre/factsheets/fs282/en/

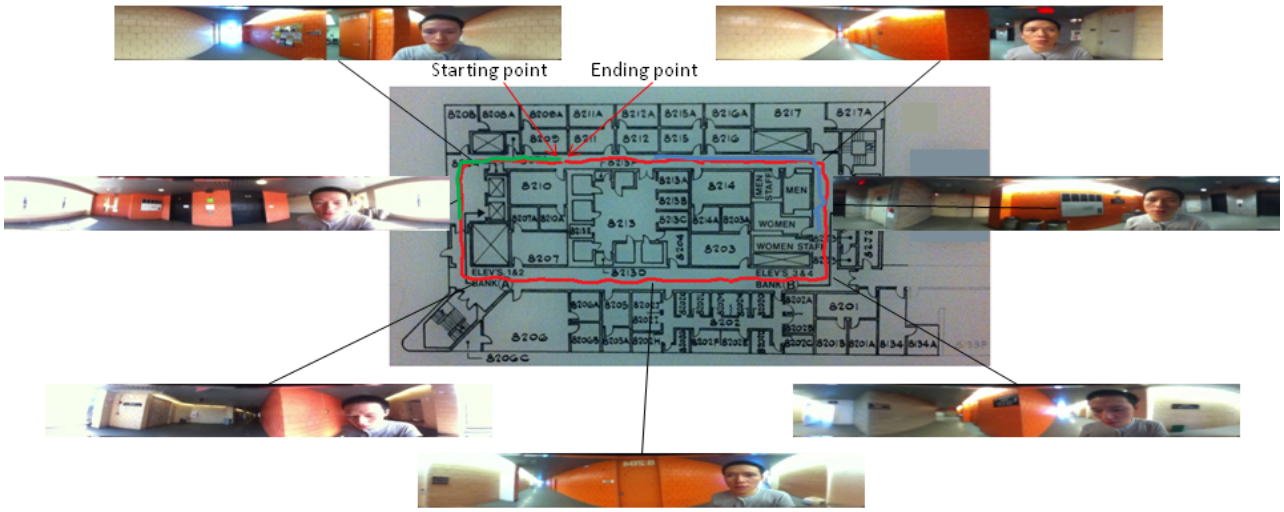


Figure 1. One of the eight floors testing environment and some sample omnidirectional images. The red line in the map is the modeled path. The blue and green lines are testing paths.

As a summary, our work has the following four major contributions: (1) A novel combination of a daily-used smart phone and a low-cost lens is used to provide an economic, portable, and robust indoor localization service for visually impaired people. (2) New omnidirectional features (omni-features) extracted from 360×82 degrees field-of-view images are proposed to represent visual landmarks of indoor positions, and then used as online query keys when a user asks for localization services. (3) A scalable and light-weight computation and storage solution is implemented by transferring big database storage and computational heavy querying procedure to the cloud. (4) Real-time query performance of 14 fps is achieved with a Wi-Fi connection by identifying and implementing both data and task parallelism using many-core NVIDIA GPUs.

This work is an extended version of our previous workshop paper [12]. We have made a number of major extensions in both algorithm design and experimental analysis so the system became much reliable and robust. First, the scale of the testing database is significantly expanded from a few hundreds of frames to a big campus building—around 50, 000 frames. Since scaling up of the database causes the server-side searching more challenging, we have designed a subspace based architecture and a multi-stream GPU paralleling algorithm (instead of a single kernel) for achieving a real-time searching performance. Second, in the workshop paper, we only provided a rough idea of how the client side algorithm and HCI should be, without detailed implementation and testing on real devices, while in this extended version, we have fully implemented the whole algorithm and all the essential HCI on iPhones; a demo of the front end app is also shown in Video A[†]. Third, to solve the tilt angle problem of the handheld camera, in this journal version we have designed and implemented a solution by detecting the built-in gyroscope angle of the phone and giving the user a vibration (or sound) hint if the phone is not held perpendicularly (with a threshold of 9 degrees). Finally, we also increase the modeling density in the modeling stage by using a multi-path sampling instead of a single path sampling for

[†]https://youtu.be/02C17A2dpWI

applications involving large rooms or very wide corridors, design and implement a 2D multi-frame aggregation method based on candidates' distribution densities for a more robust performance when the testing paths are different from the original modeling path.

The organization of the rest of the paper is as follows. Section 2 discusses related work. Section 3 explains the main idea of the proposed solution and describes the overall approach. Section 4 illustrates the image preprocessing and feature extraction procedure. Section 5 discusses the localization by indexing approach, the multi-frame aggregation algorithm as well as issues in parallel processing. Section 6 provides real data experimental results with both accuracy and time performance analysis. Finally, Section 7 gives a conclusion and points out some possible future research directions.

2. Related Work

Appearance-based localization and navigation have been studied extensively in the computer vision and robotics communities using a large variety of methods and camera systems. Outdoor localization in the urban environment with panoramic images captured by a multi-camera system (with five side-view cameras mounted on the top of a vehicle) is proposed by Murillo et al. [13]. Another appearance approach proposed by Cummins and Newman [1] is based on Simultaneous Localization and Mapping (SLAM) for a large scale road database, which is obtained by a car-mounted sensor array as well as a GPS. Kanade et al. [14, 15] localize vehicles by first driving through a route and then comparing the captured image with the image captured when the vehicle is driven to the same place for the second time. These systems deal with outdoor environment localization with complex camera systems. In our work, we focus on the indoor environment with simple but effective mobile sensing devices (smart phone + lens) to serve the visually-impaired community.

Direct 3D sensing based localization approaches are investigated by some researchers [16–18]. Microsoft Kinect or Google Project Tango device are used to construct 3D models for indoor environments. By utilizing the 3D information, the system notifies the user what should be the next move or where the user is via assistive user interface. These systems, however, require extra non-daily used devices, which make them not easy to integrate into blind people's daily life. A visual nouns based orientation and navigation system for blind people was proposed by Molina et al. [19], which aligns images captured by a regular camera into panoramas, and extracts three kinds of visual nouns features (signage, visual text, and visual-icons) to provide location and orientation instructions to visually-impaired people, using visual noun matching and PnP localization methods. In their work, obtaining panoramas from images requires several capture actions and relatively large computation resources. Meanwhile, sign detection and text recognition procedures face a number of technical challenges in a real environment, such as illumination changes, perspective distortion, and poor image resolution. In our paper, an omnidirectional lens GoPano [20] is used to capture panorama images in real-time, and only one snapshot is needed to capture the entire surroundings rather than multiple captures. No extra image alignment process such as [19, 21] is required, and no sign detection or recognition is needed.

Another related navigation method in indoor environments is proposed by Aly and Bouguet [22] as part of the Google Street View service, which uses six photos captured by professionals to construct an omnidirectional image of each viewpoint inside a room, and then estimates the camera pose and moving parameters between successive viewpoints. Since their inputs are unordered images, they construct minimal spanning tree among the complete graph of every viewpoint to select triples for

Figure 2. System diagram.

parameter estimations, which is computational intensive. In our method, since we use sequential video frames, we do not need to find such spatial relationships between images, therefore the computation cost is reduced for a real-time solution.

Representing a scene with extracted features for different purposes, such as image classification or place recognition, has been studied by many researchers [23–25]. An early work in representing and compressing omnidirectional images into compact rotation invariant features was proposed by Zhu et al. [26], where the Fourier transform of the radial principle components of each omnidirectional image is used to represent the image. Different from [26], based on the observation that an indoor environment usually includes a great number of vertical line segments, we embed these vertical line segment distribution information into a one-dimensional omnidirectional feature, and then use the Fourier transform components of these features as the representation of omnidirectional images. Another major difference is that we aim to find a user's location and orientation from each omnidirectional image, whereas in [26], only six road types are classified using a neural network based approach.

Mobile and wearable devices are cheap and ubiquitous nowadays, which accelerate the advancement of both the general computer vision [27, 28] and assistive applications. Farinella et al. [29] use Android phones to implement an image classification system with DCT-GIST based scene context classifier. Altwaijry et al. [30] apply Google Glass and develop an outdoor university campus tour guide application system by training and recognizing the images captured by Glass camera. Paisios [31] (a blind researcher) creates a smart phone HCI for the Wi-Fi based blind navigation system. Manduchi [32] proposes a sign-based way-finding system and tests the blind volunteers with smart phones to find and decode the information embedded in the color marks pasted on the indoor walls. However, there is very few research work on designing user-friendly smart phone apps (e.g., iOS app) for helping visually impaired people to localize themselves and navigate through an indoor environment.

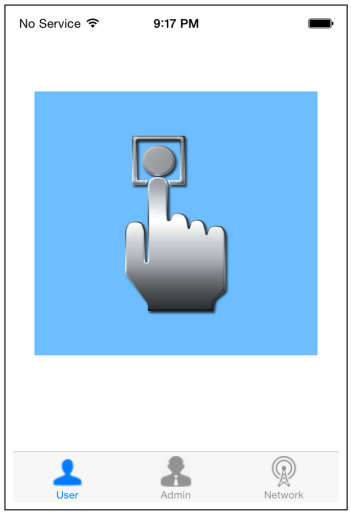


Figure 3. iOS app interfaces.

(a) **(b)**

Figure 4. Helmet-mounting and hand-held working modes.

3. System Overview

The omnidirectional assistive localization system uses a client-server architecture. The server stores the environmental databases, and has an indexing program running all the time, waiting for the queries from the client side. The client side is implemented on a smart phone as an app, such as an iOS app. The client side hardware consists of a smart phone and an omnidirectional lens, which is mounted on the phone with a case. In our implementation, we use an iPhone and a GoPano lens. A system diagram is shown in Figure 2. There are two possible ways of using this system. One is hands-free mode, as shown in Figure 4a. The camera is mounted on the top of a bicycle helmet worn by a blind user. The other is a hand-held mode, as shown in Figure 4b. Localization query action is completed by a single pressing on a big button in the middle of the smart phone's screen, as shown in Figure 3. During a usability study, we received a lot of feedback from the blind community for the project. Originally, we would like to mount the smart phone onto a helmet, so users do not need to hold phones by themselves. However, according to their feedback, the majority of blind people prefer to hold the phone by hand to avoid unnecessary attentions. Also, since the users only need to hold the smart phone for a few seconds

when a localization service is needed, they do not have to hold the phone all the way while walking. As a result, in this paper, we will mainly focus on the second working mode.

The building of the system includes two stages: the modeling stage and the querying stage, as shown in Figure 5. In the modeling stage, the system developer carries the smart phone and moves along the corridors with an even normal walking pace (0.5 m/s to 2 m/s), covering and recording the video into a database. Geo-tags (e.g., physical locations of current frames) are manually labeled for associated key frames. Motion estimation algorithms can be used in the future to ease the constraint of linear motion. To reduce the storage need, some preprocessing procedures are carried out to the images, which will be discussed in detail in Section 4. In the querying stage, a visually impaired user can walk into the area covered in the above modeling stage and take a short video clip. The smart phone extracts video features and sends them to the server via a wireless connection. The server receives the query, searches the image candidates in the database concurrently, and returns the localization and orientation result to the user.

For the modeling stage, we use all the frames of the video to compose the database in order to make full use of the visual information and obtain the highest possible localization resolution for each scene. During the labeling procedure, we select associated key frames, where landmarks (e.g., door plate, corners) are located while these frames are captured, and then labeling the rest frames in between by interpolation in the defined floor plan coordinate system. In the testing stage, we sample the testing videos by selecting one from every 5-10 frames, to reduce the query computation while still make use of the multi-frame query advantage.

Using all the pixels in images of even a short video clip to represent a scene is too expensive for both communication and computation, and the data are not invariant to environment variables such as illumination or user heading orientation. Therefore, we propose a concise and effective representation for the omnidirectional images by using a number of one-dimensional omnidirectional projections (omni-projections) for each panoramic scene. It is observed that an indoor environment often has plenty of vertical lines (door edges, pillar edges, notice boards, windows, etc.), and the distribution of these lines around the user provides unique landmarks for localization. However, for computational efficiency as well as robustness, we don't extract those line features directly; instead we embed them inside of the proposed omni-projection representations, even though these line features can be utilized in the future with a correlation-based matching method to estimate a viewer's location.

Newly extracted features of an input image are used as query keys to localize and navigate in the environment represented by the database. Because the omni-projection presentation will be different even at the same location if the smart phone is not held vertically, we develop a function in the database acquisition and testing procedure by extracting the smart phone's built-in gyroscope values, and give the developer or the user a hint if the phone is not held properly.

Many man-made structures and objects look very similar, particularly for indoor environments, so it's a challenging task for accurate localization using only single frame. We therefore adopt a multiple frame querying mechanism by extracting a sequence of omni-projection features from a short video clip, which can greatly reduce the probability of false positive indexing. When the database scales up for a large scene, it is very time consuming to sequentially match the input frame features with all the images in the database of the large scene. So we use General Purpose GPU (GPGPU) to parallelize the query procedure and accelerate the querying speed, the details of which will be described in Section 5.

Like any other image-based localization method, occlusion by moving persons and scene changes

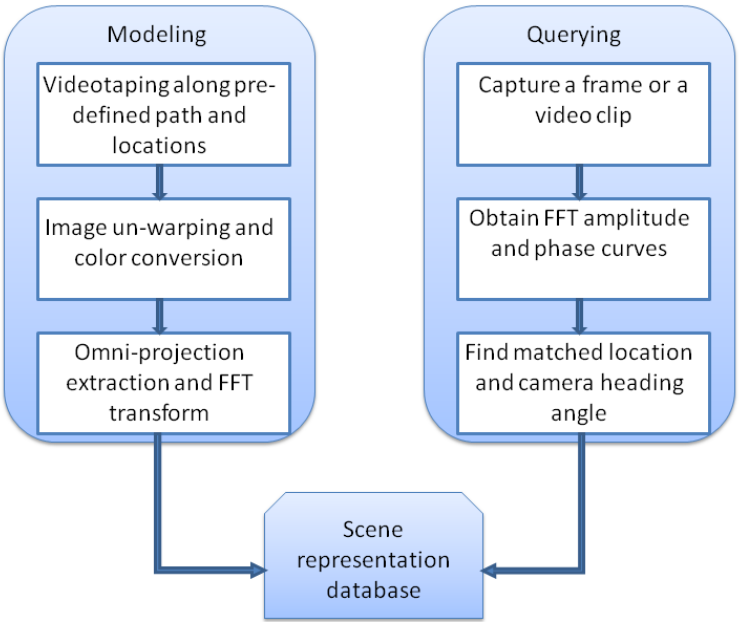


Figure 5. Work-flow of modeling and querying stages.

due to varying illuminations will influence the scene representation and thus challenge the effectiveness of localization systems. In extreme cases, if the scene representations are identical with each other (e.g., empty rooms with uniform-colored walls), or there are large self-occlusions (e.g., camera is blocked by users hand), all appearance-based image localization system will have challenging problems. However, our system has two advantages over other approaches: (1) the omni-lens has a 360×82 degrees field of view, which captures wider areas visual information and is less vulnerable to local occlusions or scene changes if they only occupy in small part of an omnidirectional image; (2) It is not a single image, but a short video sequences captured while the user is walking is utilized, so even though one of the frames has significant occlusions or changes, the rest frames can help to provide stable environment visual information for an effective localization.

4. Preprocessing and Omni-Feature Extraction

The original image frames captured by the GoPano lens and smart phone camera are fish-eye-like distorted images, as shown in Figure 6a, which has to be rectified. Figure 6c shows the un-warped image in a cylindrical projection representation, which has planar perspective projection in the vertical direction and spherical projection in the horizontal direction.

For achieving un-warping, a calibration procedure is needed to obtain all the required camera parameters. Denote the original pixel coordinate system as $X_iO_iY_i$, the un-warped image coordinate system as $X_eO_eY_e$, and the original circular image center as (C_x, C_y), then a pixel (x_e, y_e) in the un-warped image and the corresponding pixel (x_i, y_i) in the original image can be modeled as

$$\begin{cases} x_i = (r - y_e)cos(x_e \times \frac{2\pi}{W}) + C_x \\ y_i = (r - y_e)sin(x_e \times \frac{2\pi}{W}) + C_y \end{cases} \tag{1}$$

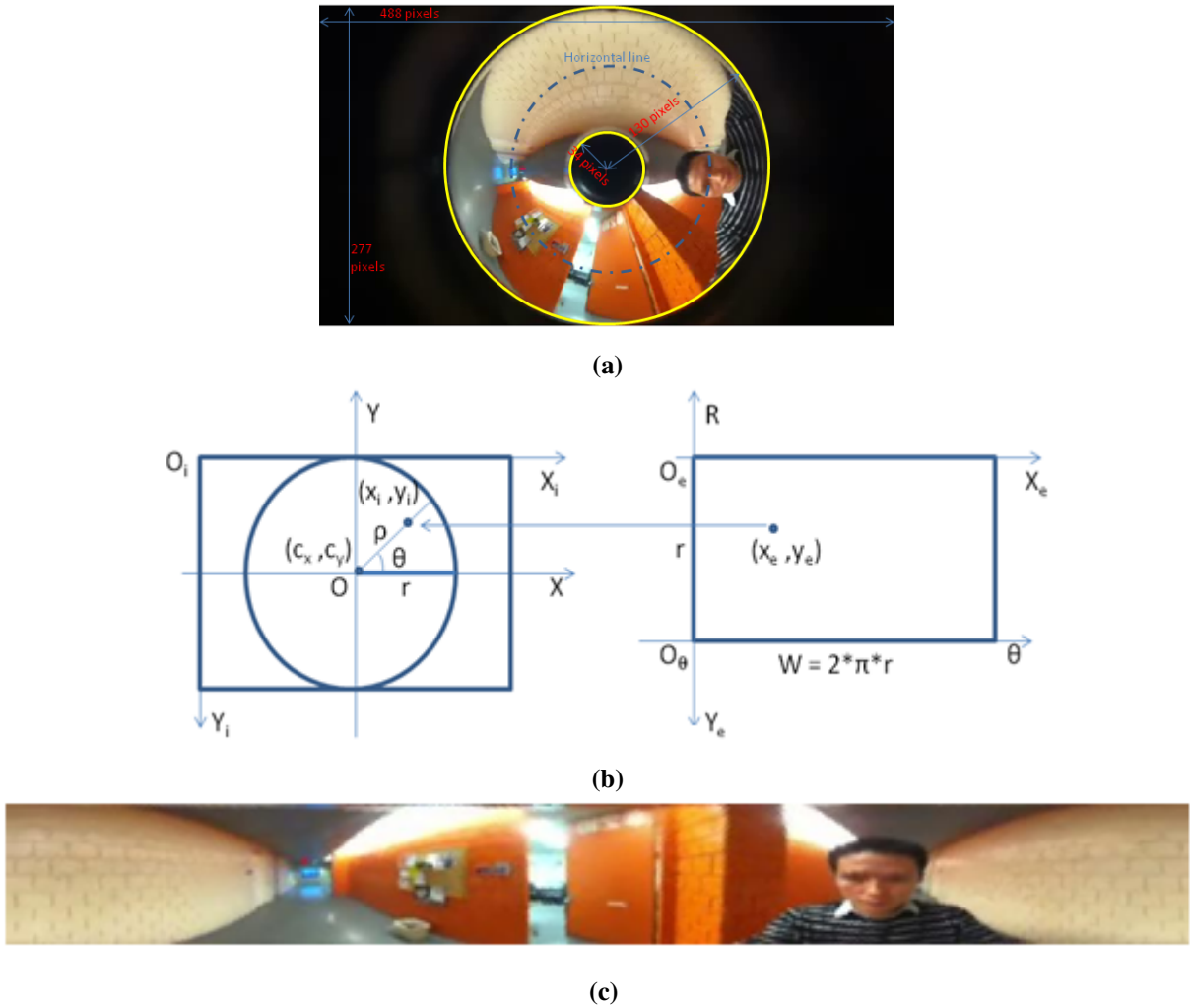


Figure 6. (a) Original video frame and its parameters; (b) Geometric relationship between the original and un-warped images; and (c) un-warped omnidirectional image.

where r is the radius of the outer circle of the original circular image—the vertical dimension (in pixels) of the cylindrical image, and $W = 2\pi r$ is the perimeter of the circle, which is turned to the horizontal dimension (in pixels) of the un-warped cylindrical image.

After obtaining the preprocessed image, we can extract one-dimensional omnidirectional feature from it.

There are many omnidirectional image calibration algorithms [33–35], but the majority of them require heavy computation, which is not practical especially if implemented on a smart phone with limited computation resources. In our work, we apply a simple but effective calibration method. We assume that the camera is held up-right and the lens' optical axis is horizontal, then we can convert the fisheye image into a cylindrical representation [36].

This un-warping process is applied to every frame in the database and all the input query frames. Since such un-warped images still have distortion in the vertical direction (radial direction in the original images) due to the nonlinearity of the GoPano lens, we perform an image rectification step using

a calibration target with known 3D information to correct the radial direction so that the projection in the vertical direction of the un-warped cylindrical images is a linear perspective projection [37]. By doing this, the effective focal length in the vertical direction is also found.

We do not directly use the pixel values of the frames to query with the database. Instead, we extract rotation invariant features (denote it as Omni-Features) from the omni-projection curves, for both images' Hue, Saturation, and Intensity (HSI) channels and their respective gradients.

According to our observation, when a visually impaired person capturing an omnidirectional image by holding the imaging system, the vertical lines usually appear in the middle of an image, while the topmost and bottommost part are ceilings and floors—such as the central black area in Figure 6a—which do not provide valid information. Therefore, we denote the middle area as the Region of Interest (ROI), and generate our omni-features by only using the ROI. Currently rectangular ROI is manually determined to be 10% of the top and bottom image area to remove useless information and also lower the memory usage as well as transmitting overload.

How to represent a scene with unique and concise features using the omnidirectional images? In our work, to represent a scene, six omni-projection curves are extracted from each corresponding cylindrical image channels. Three of them are from Hue, Saturation and Intensity (HSI) channels of the image, and the remaining three are from the gradient magnitudes of the Hue, Saturation and Intensity channels. Denote the original image of the HSI channels as $f(u, v)$, where u and v stand for the horizontal and vertical directions respectively. The gradient magnitude image of $f(u, v)$ is calculated as follows

$$|\nabla f(u, v)| = |\frac{\delta f}{\delta u} du + \frac{\delta f}{\delta v} dv| \tag{2}$$

where f can be H, S and I channels, $\frac{\delta f}{\delta u}$ is gradient in the u direction, and $\frac{\delta f}{\delta v}$ is gradient in the v direction. L_2 norm of the $(\frac{\delta f}{\delta u}, \frac{\delta f}{\delta v})$ is used to calculate the magnitude. In practice, since we mainly focus on the vertical lines—horizontal changes of the images, we only need to calculate the horizontal gradient.

We further define function g(u,v) as the image function to represent one of the six types of images (H, S, I and their gradient magnitudes, or simply gradients). The omni-projection curve $c(u)$ of the feature function $g(u, v)$ is generated by projecting the ROI of the image in the v direction:

$$c(u) = \sum_{v=0}^{H-1} g(u, v), u = 0, 1, 2, ..., W - 1 \tag{3}$$

where W and H are the image width and height of the ROI (where H is smaller than r), and $u = 0, 1, 2, ..., W - 1$ are the horizontal pixel indexes (corresponding angles from 0 to 360 degrees). As we can see the curve $c(u)$ is a one dimensional omnidirectional projection curve.

A linear normalization is applied to all the curves to turn them into the same scale and increase the robustness for the illumination changes.

With all the six curves, we store each and every one of them into environment database, and use them to compare with the new input curves for finding the optimal location for a newly input frame.

The omni-feature extraction algorithm is summarized in Algorithm 1. In the calibration step (Step 2), the camera is calibrated by finding the image center's position, the outside and inside radii of the circular images. In Step 3, a look-up table is built up between the original image *oriImage* and the undistorted image *unDistortedImage* for fast processing, using the geometric relationship between this

Algorithm 1: Omni-feature extraction.

Data: An omnidirectional video captured by an iPhone
Result: Each frame's vector representation of the omni-feature *omniVector*

```
1  initialization;
2  GoPano image calibration;
3  generate a look-up table Tab() between oriImage and unDistortedImage;
4  while hasNextFrame() do
5      for each pixel (u, v) in unDistortedImage do
6          calculate unDistortedImage(u,v) value using Tab();
7      end
8      iImage = rgb2hsi(unDistortedImage);
9      gImage = horizontalGradient(iImage);
10     pVector = verticalProjection(gImage);
11     npVector = normalize(pVector);
12     fVector = FFT(npVector);
13     omniVector = selectPrincipleComponent(fVector));
14 end
```

two images, as shown in Figure 6b. Then, for each frame, we generate the undistorted image (Step 5), convert it from RGB color space to HSI space (Step 8), and select the intensity channel *iImage*. After that, we apply gradient operation to the *iImage* (Step 9), and project the resulting image onto an omnidirectional vector (Step 10). Finally, we normalize the vector (Step 11), transfer it into Fourier domain (Step 12), and generate the vector representation of the omni-feature by selecting the principle component (the first half of the vector) of the FFT result for reducing transmitting and storage load (Step 13).

Note that our work does not directly use individual vertical lines to represent or match scene images. Instead, we use a global feature vector to represent each panoramic scene image, and then match the pre-built scene omni-feature database extracted using the same feature extraction method. Our features, however, do relate to vertical lines of the scenes, because after we get the omni-features, these line features are included. Video B[‡] shows the relationship between omni-features and vertical lines by real-time detecting vertical lines from our omni-features. Since the detecting algorithm is off the topic, we do not include it here, however, it could be used as an alternative feature-based localization method in the future.

5. Localization by Indexing

Localization service is critical to a visually impaired person's normal life, not only because it provides with the current position and orientation of a user, but also because it could supply additional information of the environment, e.g., locations of doors, positions of doorplates, which are very useful for them to judge and make decisions. The essential idea of omnidirectional imaging based localization is that we employ for each scene a unique and distinguishable feature, so each scene position is

[‡] https://youtu.be/MgGy_qt6I-Y

indexed by the omnidirectional feature. In this section, we will introduce in detail how to localize a user via indexing using our omni-features for visually impaired people.

5.1. Rotation-invariant feature extraction and rotation estimation

One question raised is that what if in the query stage, the images acquired at the same location, have a different heading direction. In this case, even though two omnidirectional images are captured at the identical location, they have different image contents. We prove that our omni-feature is rotation-invariant and thus have the ability to represent a scene no matter what heading direction is used when capturing this image up to a shifting angle.

For any arbitrary omni-projection curve $c(u), u = 0, 1, 2, ..., W-1$, if the camera rotates around the vertical axis, it will cause a circular shift of the cylindrical representation of the omnidirectional image, which then corresponds to a circular shift to the signal $c(u)$. If an omnidirectional image has a circular shift of u_0 to the right, this is equivalent to rotating the camera coordinate around z axes for $\Phi = -2\pi u_0/N_0$ [26]. Suppose the signal after a right circular shift u_0 is $c'(u)$, we have the following equation:

$$c'(u) = c(u - u_0) \tag{4}$$

Define the FFT of $c(u)$ as a_k, the FFT of $c'(u)$ as b_k, then

$$\begin{cases} a_k = \sum_{u=0}^{W-1} x(u) e^{-j2\pi ku/W}, k = 0, 1, ..., W-1 \\ b_k = a_k e^{-j2\pi ku_0/W}, k = 0, 1, ..., W-1 \end{cases} \tag{5}$$

As we can see from the second equation of Equation 5, the magnitudes of the omni-projection curves are rotation-invariant, i.e., $|a_k| = |b_k|$. This is also why we use the FFT magnitudes of the six omni-projection curves of a query frame to do indexing in the database for avoiding the heading direction problem.

Utilizing the whole amplitude curve is much more economical in terms of storage used compared with storing the whole image pixels. We further improve this by only selecting the first half of the FFT magnitudes of each curve $|a_k|$ for indexing without losing localization accuracy by taking advantage of the symmetry property of the FFT transformation for discrete signals.

How we can then find the location and heading direction using the above feature? When a new omni-feature is extracted and we want to find its corresponding scene location, the distances of the six curves between the query and each database frame are calculated. The final match is the database frame that yields the smallest L_2 distance. From our experiments, we have found that the gradient of intensity curve provides the best discrimination. Once the correct location is determined with the above approach, we can further calculate the heading direction of the new omnidirectional input. We formulate the problem of finding the rotation angle of the current image (i.e., the amount of circular shift, or the heading direction of the image) to finding the maximal value of the circular correlation function (CCF)

$$CCF(u_0) = \sum_{u=0}^{W-1} c(u) \times c(u - u_0) \tag{6}$$

where $u_0 = 0, 1, ..., W-1$. According to the correlation theorem, we can calculate $CCF(u_0)$ as

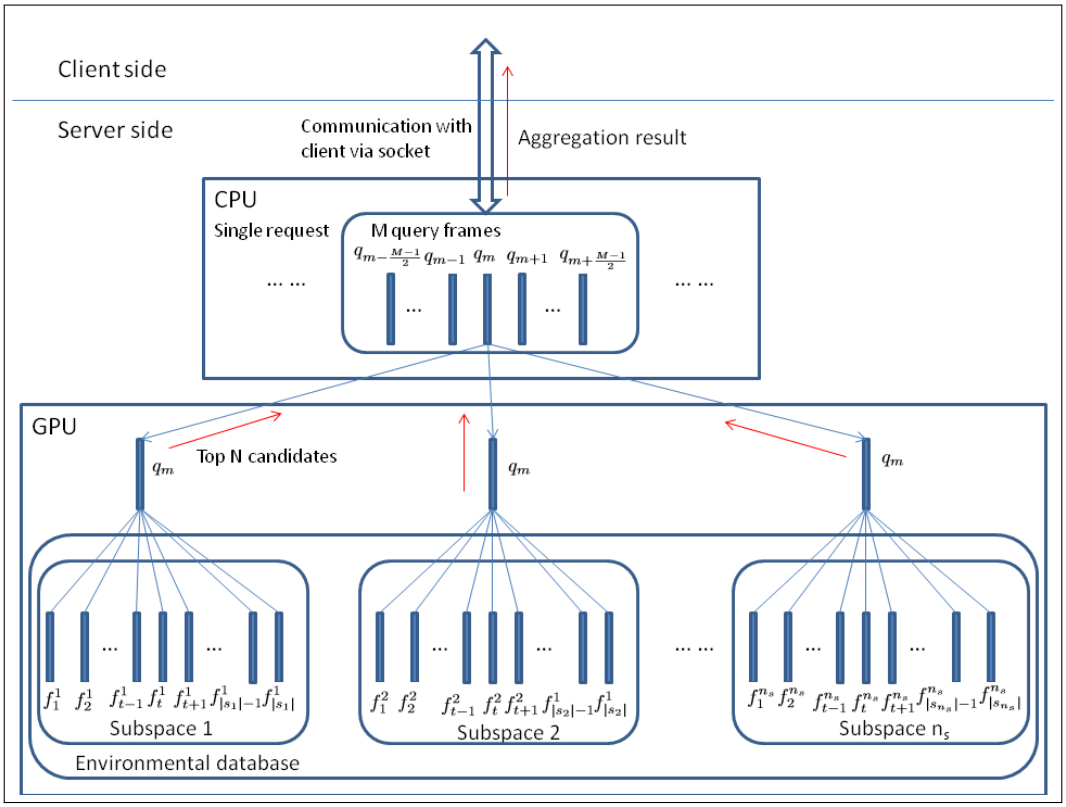


Figure 7. Parallel query with GPU in the server side.

$$CCF(u_0) = F^{-1}\{a^*(k)b(k)\} \tag{7}$$

where F^{-1} is inverse Fourier transformation operator, and * is the conjugate operator.

Now we are able to find both the location and heading direction of a new input image. There are, however, still more issues left. In the real environment, because of the existence of the noises, the localization result using one image is not always accurate and not stable. In the following subsections, we will talk about how to solve this problem by using multiple images and aggregation techniques.

5.2. Multi-frame parallel indexing and median filtering

To increase the robustness of the localization algorithm, using a single input omnidirectional image is not sufficient; we need to input a sequence of images to query the environment database for accurate localization. Each input image returns an optimal location, and all the returned locations are aggregated to generate a final location result.

When multiple frames are utilized, querying them with normal CPU is not efficient, therefore, we utilize GPU along with corresponding parallel algorithms to ensure a real-time performance. Figure 7 shows a diagram of our GPU parallel searching algorithm on the server side. When a client sends a localization request of a frame q_m, it actually sends $\frac{M-1}{2}$ frames before the frame q_m, and $\frac{M-1}{2}$ frames after the frame q_m to the server (e.g., $M = 3$ or 5). The CPU of the server will copy each frame's feature to the GPU, and receive the top N candidates calculated by the GPUs for each frame after processing.

The environmental database is also divided into n_s subspaces (e.g., each floor of a building's environmental data can be stored as separate subspaces), and the n_i subspace has $|n_i|$ modeled frames. All the subspaces therefore can be searched in parallel with GPU, instead of searching all the subspaces sequentially with a single CPU.

Take the frame q_m as an example. The frame q_m is compared with all the frames f_t^i (t = 1, 2, ..., $|n_i|$) within each subspace i (i = 1, 2, ..., n_s). The same searching procedure is carried out to all frames also. Finally, all the returned candidates of the M query frames are aggregated and return to the client via TCP/IP socket.

To aggregate the return candidates, we can take advantage of the fact that both the query frames and the database frames are sequentially indexed, so we use a simple median filtering approach for the temporal aggregation.

Algorithm 2: GPU multi-stream paralleling indexing and median filtering.

Data: M query frames received from CPU
Result: Location candidates *FinalResult* calculated by GPU

```
1  initialization;
2  selectNearbyFrames();
3  for each selected frame q_m do
4      for each CUDA stream do
5          memoryCopyFrmoCPUtoGPU(q_m);
6          for each CUDA thread do
7              calculateDistance(q_m, f_i);
8          end
9          synchronizeThreads();
10     end
11     synchronizeStream();
12     Results = selectTopCandidates();
13     memoryCopyFromGPUtoCPU(Results);
14     FinalResult = medianFiltering (Results);
15 end
```

The whole process could have four levels of parallelisms. First, we divide the search database into multiple subspaces, for example, each floor data is counted as one subspace, and all the subspaces can be parallelly searched with CUDA streams. Second, within each subspace, we process all the frames of an input query video clip in parallel. Third, we use multiple threads to compare each input frame with multiple database frames simultaneously, rather than comparing with them one-by-one. Fourth, some of the operations in obtaining rotation-invariant projection curves, such as the Fourier transform algorithm can take advantage of the parallel processing. For doing this, the whole original omni-projection curves will be sent from the front end to the server, which is still very efficient in communication due to their low dimensionality.

We formalize our algorithm of median-filter-based parallel indexing and show it in Algorithm 2. In the initialization step, the server receives the data from the client and prepare for querying. In Step 2, for each query, the server will not only select the current testing frame, but also multiple other frames

near the testing frame. Then, all the selected frames are copied to each CUDA stream one by one (Step 5), where the database of each subspace are stored and computation is taken care of concurrently. By utilizing multiple stream, we can process the search work within each subspace, for example each floor of a tall building, concurrently, instead of searching them sequentially. This is the first level of parallelism.

Within each stream, there are multiple threads employed for the searching work, e.g., 32×256 threads, and each thread is responsible for the comparison of the querying frame and database frame (Step 7). So instead of using sequential algorithm with CPU only and comparing the query frames and database frame one-by-one, we can accelerate the comparison work by many times via simultaneously comparing the input frames and the database frames. This step implements the second and third level of parallelism since streams are executed simultaneously, and all the threads within each stream are also carried out in parallel. We could further optimize our implementation with more parallel algorithms, such as the ones for the fourth level of parallelism to achieve more fast performance when needed. CUDA synchronization is called to make sure all the tasks within each thread (Step 9) and all the work in each stream (Step 12) are finished before the program moves to the next step. Finally, the GPU collects the returned candidates, which could be used for the median-filter-based aggregation (Step 14).

5.3. 2D multi-frame aggregation based on candidates' densities

For majority of the indoor environments, such as normal corridors, rooms, the environment is modeled by walking through it once, and the above median-filter-based aggregation algorithm works well. However, for the environment with wide corridors or large rooms, modeling with a single traverse is not sufficient and we need multiple traversals (paths) to be able to cover the complete area and provide accurate result. Also, for a multi-frame based localization query, the query frame is sent to the server together with a number of frames before and after it. After receiving a large amount of position candidates for each query frame, a more sophisticated aggregation algorithm is needed to integrate all these candidates together before we deliver a finalized location coordinate to the user. In this subsection, we will talk about these two problems in detail.

We demonstrate the application of the system for the areas such as corridors or rooms, which are large, by increasing the densities of the environmental modeling with multiple paths of video capture. Instead of capturing a video on only one path, we use multiple parallel paths to build the environment model. After that, we correlate the paths with the physical space coordinates by performing associated frame labeling and interpolation. Then, the final location is obtained (using Algorithm 3) by aggregating the possible candidates in the 2D coordinate plane.

If there is no noise, in an ideal case, for any input testing frame, all the returned candidates from the server should be at the exact the same position where the testing frame was captured, since the testing frame and the modeling frames (who generate the candidates) shall capture identical environmental information. However, because of various reasons (noise, scene similarity, etc.), some of the candidates may be disturbed and are far away from the ground truth position, while the majority of the candidates cluster around the ground truth area. In this paper, according to the above observation, we have designed and implemented a 2D multi-frame aggregation algorithm by utilizing the densities of

Algorithm 3: 2D multi-frame localization candidates aggregation.

Data: Location candidates on the floor plan coordinate system
Result: Aggregated final location coordinate $FinalPosition$

```
1  initialization;
2  for each candidate C_i do
3  |  Accumulate locationDistriArray(C_i);
4  end
5  rankLocationDistribution(locationDistriArray);
6  TopC = findTopDensityArea(locationDistriArray);
7  for index < NumOfTopC do
8  |  if topDensityArea(index) > NumOfAllCandidates × tolerPer then
9  |  |  FinalPosition = topDensityArea(index);
10 |  |  break;
11 |  end
12 end
```

the candidates' distribution, and calculate the most likely finalized location estimation.

The algorithm is shown in Algorithm 3. In the initialization step, the candidates' indexes in the modeling frames are mapped to their actual floor plan 2D coordinates by checking the geo-referenced mapping table pre-generated while modeling the environment.

Then in Step 2, we count each tile's candidates number by a 2D binning operation, and obtain the query frame's location distribution array *locationDistriArray*. We sort this array in a descending order in Step 5, and obtain the first $TopC$ number (currently $TopC = 10$) of tiles in Step 6.

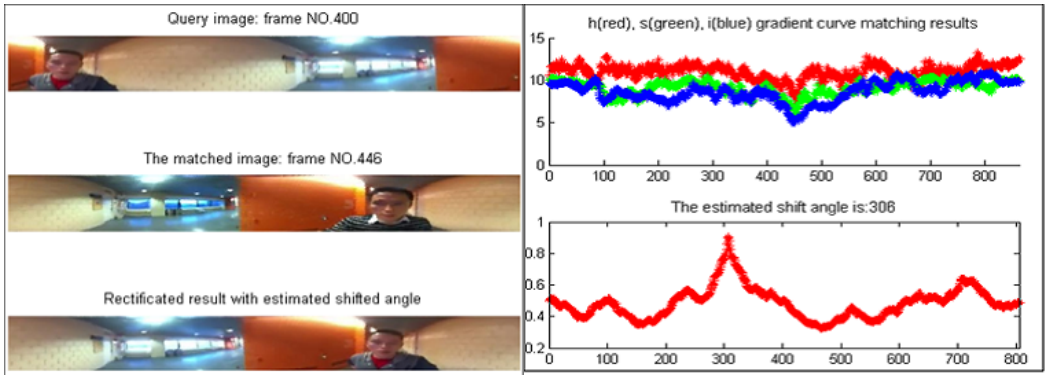
In a perfect case, the top one tile of these $TopC$ candidates should be the best estimated result since it has the largest amount of candidates dropped in. This is, however, not always the case, because the tile with the most candidates may be a false positive estimation, and its nearby tiles may have very few candidates.

To increase the robustness of the estimation, in Step 7, we set a tolerant circle around each tile, with some radius (in our experiment, this radius is 3 meter.). Afterward, we count the total number of candidates within this circle. If the amount is greater than a threshold percent— *tolerPer* (e.g., $tolerPer = 20\%$) of the total amount of candidates, the corresponding tile is estimated as the final position, and is returned to the user via the sockets.

We demonstrated the application of the system where the areas (corridors or rooms) are large, by increasing the densities of the environmental modeling with multiple paths of video capture, the details of which is shown in Section 6. Instead of capturing a video on only one path, we use multiple parallel paths to build the environment model. After that, we correlate the paths with the physical space coordinates by performing associated frame labeling and interpolation. Then, the final location is obtained (using Algorithm 3) by aggregating the possible candidates in the 2D coordinate plane.

6. Real Data Experiments: Accuracy and Time Performance

A number of experiments are carried out for testing the accuracy and time performance of the omnidirectional-vision-based localization approach, the details of which are provided in the following



(a) **(b)**

Figure 8. (a) An example of a query image and its matching result in a database; (b) the matching scores with database frames and the estimated heading differences between the query and database frames.

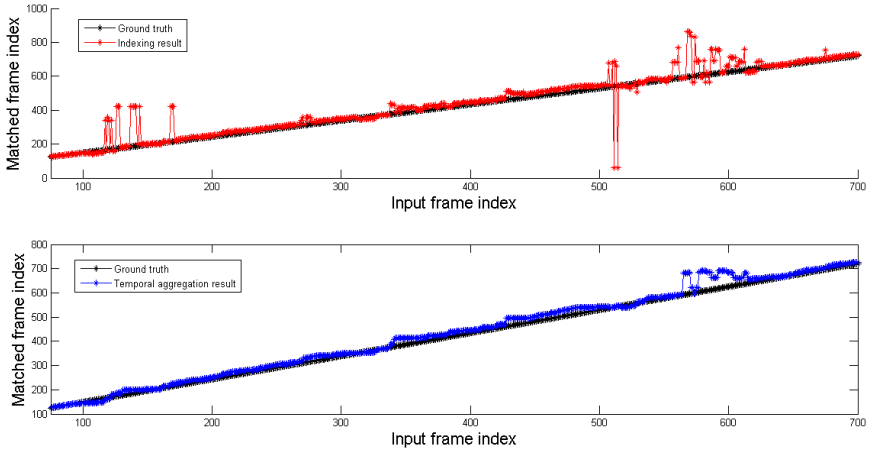


Figure 9. Matching results of all the test database frames without (top) and with temporal aggregation (bottom).

subsections.

6.1. Comparison of single-versus-multiple frame indexing

In the first experiment, we compared the accuracy of single-versus-multiple frame indexing with a small database of 862 frames modeling the environmental with one single path. One example of indexing a new query frame is shown in Figure 8. Figure 8a shows the input frame, the target frame, and the shifted version of the input image after finding the heading angle difference. In Figure 8b, the first plot shows the searching results of the input frame with all the frames in the database using hue (red), saturation (green) and intensity (blue). We found that the information of intensity feature performs the best. The circular correlation function curve is shown at the bottom, showing that the heading angle difference can be found after obtaining the real match.

Because different scenes may have very similar omni-projection curve features, a query with only

one single frame may cause false matches, as shown in Figure 9 (top). In this figure, the horizontal axis is the index of input frames (of a new sequence) and the vertical axis is the index of database frames (of the old sequence). Both sequences cover the same area but are captured at different dates. The black curve shows the ground truth data of matching results, and the red curve shows the matching results by our system. As we can see, there are a few obvious mismatches around frame 150, 500, and 600 due to the scene similarities.

This leads us to design a multiple-frame approach: if we use a short sequence of input frames instead of just one single frame to perform the query, a temporally consistent match result for all the input frames will yield a much more robust result.

Figure 9 (bottom) shows the testing results after the temporal aggregation. In this figure, for every frame, the querying results of its nearby 25 frames are aggregated and the median index value is used as the final result. We take advantage of the fact that both the query frames and the database frames are sequentially indexed, so we use a simple median filter of the 25 indexing results to obtain the temporal aggregation result. By a simple calculation, the average indexing error reduces to 14.7 frames with the simple temporal aggregation, from 25.3 frames with a single frame indexing. These correspond to 0.29 m versus 0.51 m distance error in space. As we can also see from the curve, temporal aggregation has corrected the very obvious mismatches and generated more robust results.

6.2. Impact of tilt angles of the camera

Previous idea assumes that while capturing the omnidirectional image, the smart-phone and the omnidirectional lens are held vertically in both the environmental modeling and online querying procedure. Even though this assumption is easy to satisfy in most cases if we provide the visually impaired people a basic training before using the localization system, there are situations where the smart-phone may not hold perfectly vertically.

In this subsection, we further analyze the impact of the holding tilt angle problem quantitatively, and provide an effective solution via utilizing the smart-phone IMU data for detecting the tilt angle and using voice feedback to warn the users.

To ensure that the smart phone is held (almost) vertically in both the modeling and testing procedures, the tilt-correction function we have developed uses a simple, but effective method to detect the current smart phones tilt angle by checking the built-in gyroscope sensor. If the tilt angle is larger than a threshold (9 degrees in current setting), the smart phone will vibrate and hint the user to adjust the position, as also shown in Video A[§]. With a little bit of training, the users can quickly adapt the correct way of holding the smart phone. In the future, we can even use the detected tilt angles to rectify the images if the angles are relatively small (below or close to the threshold).

If the camera lens is not pointing perpendicularly up, the omnidirectional features will change a lot, thus even two images captured in the same position have very different omni-features. Figure 10 shows the matching results with and without tilt-correction function on, where the black lines in both curves show the ground truth locations in terms of the frame index, the red line on the top plot is the result with tilt-correction, and the blue line in the bottom plot is the result without tilt-correction. We can see from the plots that the accuracy increases significantly after the correction.

[§]https://youtu.be/02C17A2dpWI

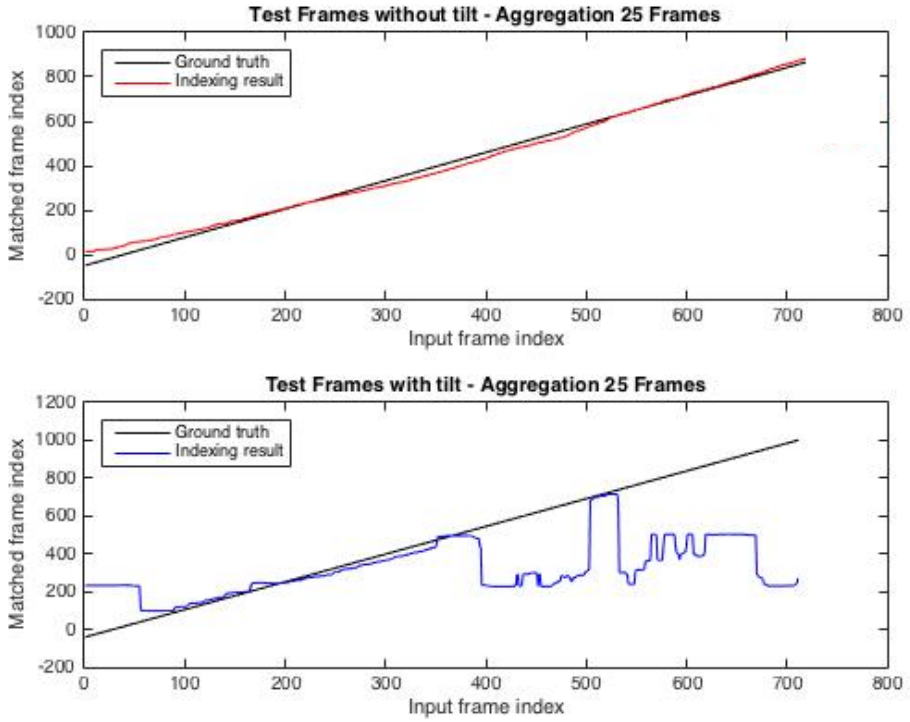


Figure 10. Matching result with and without tilt-correction function.

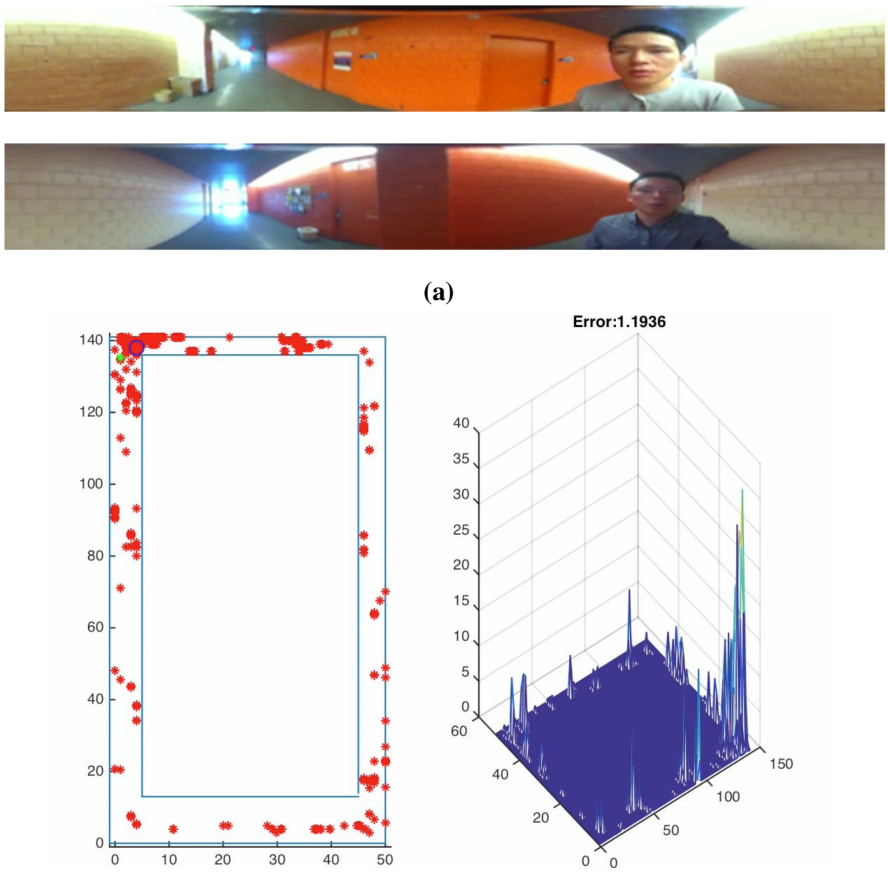
6.3. Localization in wide areas: 2D aggregation

To provide localization service in large areas, e.g., large rooms or wide corridor, single path modeling is not sufficient when the testing path is far away from the modeling path. In the third experiment, we model a whole floor with multiple paths (5 parallel paths), and test our Algorithm 3 for 2D aggregation result.

We first capture video sequences of the entire floor along these five paths as the training databases, one of which is shown in red line in Figure 1. We then capture two other video sequences as the testing data sets, as shown in the blue and green lines, respectively. Some sample omnidirectional images used in the modeling process are also shown around the map in Figure 1, as well as their geo-locations attached to the floor map.

For each testing frame, we query the entire training database and find the most similar stored features with the smallest Euclidean distance. Many scenes in the environment may be very similar, for example, the two images in Figure 11a, so the returned feature with the smallest distance may not necessary be the real match. We then select the top N (N = 15 in this experiment) features as the candidates. Also, to solve the scene similarity problem, for each querying frame, we not only use itself as the query key, but we also use M-1 (M = 11) other frames, (M-1)/2 before and (M-1)/2 after this frame, as the query keys to generate candidates. We test all these frames within all the P (P = 5) paths, the frames of which are all labeled on the floor plan coordinate system, and retrieve all the candidates. Consequently, there are total $N \times M \times P$ candidates.

In Figure 11b, the red stars are the 825 location candidates of a query frame and its related 10 frames. Note that several frames may return the same location, so each red star may represent more than one candidate. In the left image of Figure 11b, the coordinate system has the same scale as the



(b) Aggregation visualization

Figure 11. (a) Two examples of very similar scenes; (b) Matching results of a frame and its neighbors against a multi-path database.

floor plan. Each unit in both horizontal and vertical directions corresponds 30 cm in the physical space. The green star shows the ground truth location and the blue circle shows the final estimation of the testing frame. The distribution of the candidates is also illustrated in the right plot. In the right figure of Figure 11b we use the height of each point to represent the number of candidates falling into this position. Since all the modeling frames captured near the testing frame have similar contents and thus have alike features, the majority of the candidates drop near the location where the testing frame is captured. In this example, as shown in Figure 11b, the testing frame's ground truth location is on the top left corner of the floor plan (the green star), as we can see, the majority of the candidates drop on the top corner area. The rightmost corner in the right figure of Figure 11b also shows that the number of candidates (representing by the height) near the ground truth position is larger than the rest of places. There are also some false positives distributed around the whole floor as shown in both left and right figure. A short video clip can be found in Video C[¶] to visualize the robustness of the 2D aggregation algorithm.

After the aggregation of all the candidates completes, we obtain the final candidate (the blue circle in Figure 11b), and use voice feedback to notify the user. The number on the top right of Figure 11b

[¶]https://youtu.be/gPGGcAfFsvY

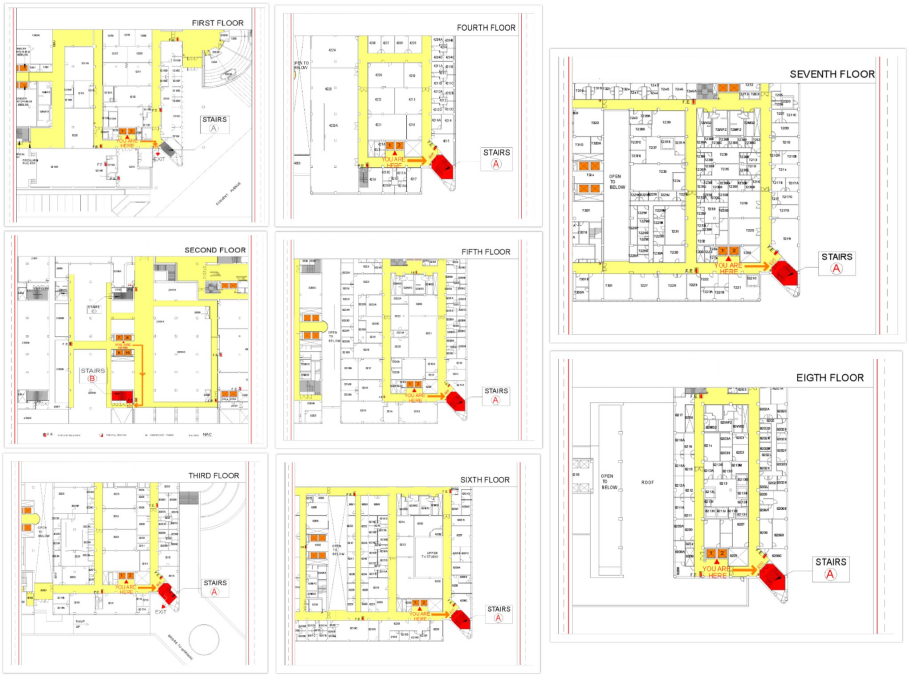


Figure 12. Floor plans of an eight-floor building.

shows the estimated error in meters.

6.4. Time performance experiment

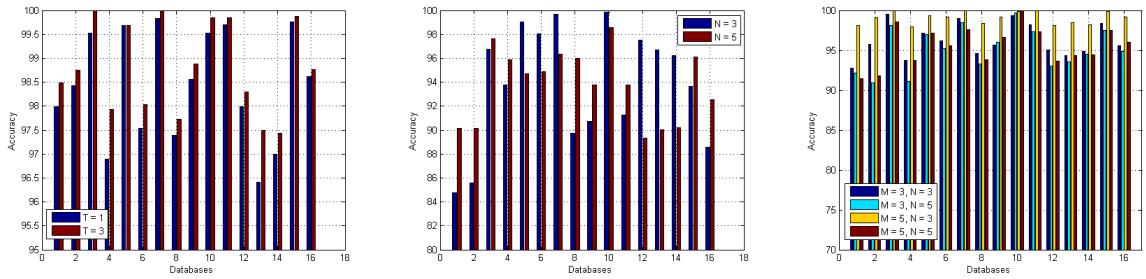
To show the time performance of the approach for a larger scale scene, in the fourth experiment, we extended the database from one floor of a campus building to the entire eight floors of the building. The floor plan of the building is shown in Figure 12.

We held an iPhone mounted with a GoPano lens, and walked through each floor while recording the video at a normal walking pace. It took 32 minutes and there were 47617 frames in total for all the eight floors. Each floor (except the second floor) has an area of around 40 meters by 14 meters (around 560 square meters). We extracted features using the method illustrated in Figure 5, and stored each floor's features into separate subspaces for later query operations.

As we discussed before, the hand-held working mode, instead of the head-mounted mode, is preferred by the users. Even though in this mode, the user's face is within the view of the camera, since it only occupies a small amount of omnidirectional images (less than 10 percent), and we use multiple images based query in the localization algorithm, it does not have significant influence on the generated features and the final localization error.

In the testing process, we selected one frame in every 5 frames for the testing, and constructed a testing data set. The total amount of testing frames is 9517 frames. These test frames are excluded from the training database so the testing frames and the database frames are not overlap with each other. We tested the time performance as well as the accuracy of our system at this campus building scale, and explored the performance of different aggregation results on different error tolerance threshold.

Currently, we assume we do not know the users position in order to make the application more general to all the cases. Applying some prior knowledge, such as initial positions, or using temporal



(a) **(b)** **(c)**

Figure 13. (a) Single-frame indexing (M = 1), and single return (N = 1); (b) Single-frame indexing (M = 1) and a 3-frame indexing tolerance (T =3); (c) Finding the best numbers for indexing frames and returns when the indexing tolerance T = 3.

information, will have some benefits, for example, reducing the searching database size, and improving time or accuracy performance, but at the same time it lowers down the usability of the system, since the blind people need to ask others for the initial position, and reduces their independence. In this work, we do not apply prior knowledge when querying the database.

Relating each frame with the floor plan, we can build a one-to-one mapping between the database frames and the walking path in the floor plan. Currently, only the frame's location within the floor plan is tagged, but it is also possible to relate the frames with more useful information, like office number, signs, etc. By checking the mapping table and the error tolerance, we can estimate the localization error in terms of real geo-location distance. For this testing experiment, we used the difference between the original frame's index and the estimated index to evaluate the localization accuracy. If the difference of the estimated index and the ground truth index were below a threshold, we labeled this query as accurate, as shown in Equation 8.

$$|I_{original} - I_{estimated}| \leq T_{threshold} \tag{8}$$

where $I_{original}$ is the original frame index number, and $I_{estimated}$ is the best matching result returned by GPU.

For each testing frame, we returned the top N (N = 3 or 5) candidates. To increase the robustness of the results, we used multiple frames nearby the current frame to query the database. Assume the number of querying frames is M (M = 3, 5), we use additional $\frac{M-1}{2}$ frames before the testing frame and $\frac{M+1}{2}$ frames after the testing frame to test the database. For every query, there are $N \times M$ candidates, and we use median value aggregation to find the best result. Typically, the few false results are outliers among a group of correct candidates. Therefore using a median filter among the $N \times M$ candidates, can remove the few outliers among the correct indexes values. Figure 13 shows the experimental results using the parallel searching strategies and the median-filtering based aggregation approach. In Figure 13a, we tested the query accuracy with different threshold $T_{threshold}$ with the single-frame indexing (M=1) and a single return for each query (N=1). When the error tolerance threshold (T) increases from 1 frame (the blue bars) to 3 frames (the red bars), the accuracy increases on all the databases: on averaged the correct rate was increased from 98.43% to 98.82%. This shows that for over 50,000 images of the 8-floor building, the scene mostly doesn't repeat itself.

In Figure 13b we tested various number of returns (N = 3 and 5) when doing single-frame query

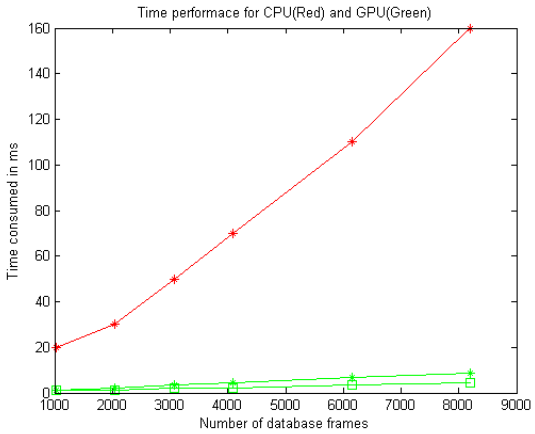


Figure 14. Time usage with and without GPU acceleration: Red without GPU; Green with GPUs. Curves with squares are experiments using 2048 threads while curves with asterisks are experiments using 1024 threads.

with a 3-frame indexing tolerance. We use the median value aggregation strategy to the N returned candidates for each query. When the number of candidates increase, as shown in Figure 13b—the number of candidates increase from 3 to 5—only half the databases' accuracy increase. Further, to find the best accuracy performance, we tested a series of combination of the number of indexing frames M and the number of top returns N for each query, as shown in Figure 13c, and the results show that when $M = 5$, and $N = 3$, the system achieves the best accuracy: the average successful rate is 99.06% when the indexing error tolerance is 3 frames (T=3).

If using only a single CPU to search sequentially, the amount of time consumed would increase proportional to the number of frames in the database. Therefore we used GPUs to do the query in parallel, so that we can search all the frames and compare an input frame to multiple database frames at the same time, which will greatly reduce the time used. Figure 14 shows the time used with and without many-core GPUs for a database with the number of images changed from 1000 frames to 8000 frames. In the single CPU version, the time spent increases from 20 ms to 160 ms, whereas using a many-core GPU (Kepler K20 chip), the time is reduced by 20 times (from 1.13 ms to 8.82 ms, for databases of 1000 to 8000 frames). Note that this test only has a database with a few thousand frames. With a larger database of an indoor scene, the time spending on a single CPU will be prohibitive, whereas using multi-core CPUs/GPUs, the time spending can be greatly reduced.

Experiments on the database of the eight-floor building demonstrated a real-time response (14 fps). Each query uses around 70 ms, among which about 40 ms is consumed by the GPU searching, and 30 ms is consumed by the socket communication. Note the current experiments have not fully used the capacity of the server. Since we can apply tens of thousands of threads in one or multiple GPUs, the acceleration rate has potential to improve.

7. Conclusion and Discussion

In this paper we proposed real-time indoor assistive localization system with mobile omnidirectional vision and cloud GPU acceleration for helping visually impaired people to localize and navigate

indoor environments. We utilize a smart phone with an omnidirectional lens as the front-end and a GPU-enabled high performance server as the back-end to ensure the portability of the user part and also take advantage of the huge storage as well as the high computation power of the server part. New omnidirectional features (omni-features) extracted from 360 degrees field-of-view images are proposed to represent visual landmarks of indoor positions, and then used as on-line query keys when a user asks for localization services. An image indexing-and-retrieving mechanism, along with multi-paths modeling and candidates' density based aggregation algorithms are employed to provide accurate locations to the users. Real-time query performance of 14 fps is achieved with a Wi-Fi connection by identifying and implementing both data and task parallelism using many-core NVIDIA GPUs.

There are a few promising future work. First, this omnidirectional vision based localization solution requires an environmental modeling before the services can be delivered to the blind people. So how we can effectively and efficiently find appropriate modeling paths for large scale environments such as a cluster of buildings using existing knowledge like architecture drawings? Second, vision based indoor localization solution will have challenges if the environmental scenes repeat themselves or having saturated illumination in extreme cases. How we can integrate other modality, such as Bluetooth or Wi-Fi to provide a robust solution when scenes are very similar? Third, since labeling the environmental model with the geo-information involves tremendous efforts, we can develop semi-automatic tools for labeling, and also get more feedback from the blind community to determine what information is more interested to them to store in the database. It would also be interesting to integrate other components of a navigation system, such as path planning or obstacles avoiding, with this localization part and create systematic overall evaluation criteria as benchmarks for furthering this research, which is our ongoing work.

Acknowledgements

The authors would like to thank the US National Science Foundation (NSF) Emerging Frontiers in Research and Innovation (EFRI) Program for the support under Award No. EFRI 1137172. The authors are also grateful to Ms. Barbara Campbell and her colleagues of the New York State Commission for the Blind (NYSCB) for their comments and suggestions in the use of the iPhone mobile vision system for assistive navigation.

Conflict of interest

The authors declare no conflict of interest in this paper.

References

1. Cummins M, Newman P (2011) Appearance-only slam at large scale with fab-map 2.0. *Int J Robot Res* 30: 1100-1123.
2. Davison AJ, Reid ID, Molton ND, et al. (2007) Monoslam: Real-time single camera slam. *IEEE T Pattern Anal* 29: 1052-1067.
3. Di Corato F, Pollini L, Innocenti M, et al. (2011) An entropy-like approach to vision based autonomous navigation. *IEEE T Robotic Autom* 1640-1645.

4. Rivera-Rubio J, Idrees S, Alexiou I, et al. (2013) Mobile visual assistive apps: Benchmarks of vision algorithm performance. In: *New trends in image analysis and processing–iciap* Springer 30-40.
5. Liu Z, Zhang L, Liu Q, et al. (2017) Fusion of magnetic and visual sensors for indoor localization: Infrastructure-free and more effective. *IEEE T Multimedia* 19: 874-888.
6. Tang H, Tsering N, Hu F (2016) Automatic pre-journey indoor map generation using autocad floor plan. *J Tec Pers Disabil* 4: 176-191.
7. Ravi N, Shankar P, Frankel A, et al. (2006) Indoor localization using camera phones. In: *IEEE Workshop on Mobile Computing Systems and Applications* IEEE 49.
8. Tang H, Tsering N, Hu F (2016) Automatic pre-journey indoor map generation using autocad floor plan. In: *31st Annual International Technology and Persons with Disabilities Conference* San Diego, CA, U.S.A.
9. Kulyukin V, Gharpure C, Nicholson J, et al. (2004) Rfid in robot-assisted indoor navigation for the visually impaired. *Intell Robot Syst* 2: 1979-1984.
10. Cicirelli G, Milella A, Di Paola D (2012) Rfid tag localization by using adaptive neuro-fuzzy inference for mobile robot applications. *Ind Robot* 39: 340-348.
11. Legge GE, Beckmann PJ, Tjan BS, et al. (2013) Indoor navigation by people with visual impairment using a digital sign system. *PloS one* 8.
12. Hu F, Zhu Z, Zhang J (2014) Mobile panoramic vision for assisting the blind via indexing and localization. In: *Computer Vision-ECCV 2014 Workshops* Springer 600-614.
13. Murillo AC, Singh G, Kosecka J, et al. (2013) Localization in urban environments using a panoramic gist descriptor. *IEEE T Robot* 29: 146-160.
14. Kume H, Suppé A, Kanade T (2013) Vehicle localization along a previously driven route using an image database. *J Mach Vision Appl* 177-180.
15. Badino H, Huber D, Kanade T (2012) Real-time topometric localization. *IEEE Int Conference Robotic Autom (ICRA)* 1635-1642.
16. Hu F, Tsering N, Tang H, et al. (2016) Indoor localization for the visually impaired using a 3d sensor. *J Technol Pers Disabil* 4: 192-203.
17. Lee YH, Medioni G (2014) Wearable rgbd indoor navigation system for the blind. In: *Workshop on Assistive Computer Vision and Robotics* IEEE 493-508.
18. Hu F, Tsering N, Tang H, et al. (2016) Rgb-d sensor based indoor localization for the visually impaired. In: *31st Annual International Technology and Persons with Disabilities Conference* San Diego, CA, U.S.A.
19. Molina E, Zhu Z (2013) Visual noun navigation framework for the blind. *J Assist Technol* 7: 118-130.
20. GoPano (2016) Gopano micro camera adapter. Available from: Available from: `http://www.gopano.com/products/gopano-micro`.
21. Hu F, Li T, Geng Z (2011) Constraints-based graph embedding optimal surveillance-video mosaicing. *Asian Conf Pattern Recogn (ACPR)* 311-315.

22. Aly M, Bouguet JY (2012) Street view goes indoors: Automatic pose estimation from uncalibrated unordered spherical panoramas. *Workshop Appl Comput Vision (WACV)* 1-8.
23. Oliva A, Torralba A (2001) Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int J Comput Vision* 42: 145-175.
24. Farinella G, Battiato S (2011) Scene classification in compressed and constrained domain. *IET Comput Vision* 5: 320-334.
25. Xiao J, Ehinger KA, Oliva A, et al. (2012) Recognizing scene viewpoint using panoramic place representation. *Comput Vision Pattern Recogn (CVPR)* 2695-2702.
26. Zhu Z, Yang S, Xu G, et al. (1998) Fast road classification and orientation estimation using omni-view images and neural networks. *IEEE T Image Process* 7: 1182-1197.
27. Sun W, Duan N, Ji P, et al. (2016) Intelligent in-vehicle air quality management: A smart mobility application dealing with air pollution in the traffic. In: *23rd World Congress on Intelligent Transportation Systems* Melbourne Australia.
28. Ma C, Duan N, Sun W, et al. (2017) Reducing air pollution exposure in a road trip. In: *24rd World Congress on Intelligent Transportation Systems* Montreal, Canada.
29. Farinella G, Rav D, Tomaselli V, et al. (2015) Representing scenes for real-time context classification on mobile devices. *Pattern Recogn* 48: 1086-1100.
30. Altwaijry H, Moghimi M, Belongie S (2014) Recognizing locations with google glass: A case study. *IEEE Winter Conf Appl Comput Vision (WACV)* Colorado.
31. Paisios N (2012) Mobile accessibility tools for the visually impaired [dissertation]. New York University.
32. Manduchi R (2012) Mobile vision as assistive technology for the blind: An experimental study. In: *The 13th International Conference on Computers Helping People of Special Needs (ICCHP)* Springer.
33. Scaramuzza D, Martinelli A, Siegwart R (2006) A toolbox for easily calibrating omnidirectional cameras. *Intell Robot Syst* 5695-5701.
34. Nayar SK (1997) Catadioptric omnidirectional camera. *Comput Vision Pattern Recogn* 482-488.
35. Kang SB (2000) Catadioptric self-calibration. *Comput Vision Pattern Recogn* 1: 201-207.
36. Scaramuzza D, Martinelli A, Siegwart R (2006) A flexible technique for accurate omnidirectional camera calibration and structure from motion. *Int Conf Comput Vision Syst* 45-45.
37. Zhu Z, Rajasekar KD, Riseman EM, et al. (2000) Panoramic virtual stereo vision of cooperative mobile robots for localizing 3d moving objects. *Omnidirectional Vision* 29-36.

AIMS Press

© 2017, the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)