Mobile Panoramic Vision for Assisting the Blind via Indexing and Localization

Feng Hu[1(✉)] , Zhigang Zhu[1,2], and Jianting Zhang[1,2]

[1] Department of Computer Science, The Graduate Center, CUNY, New York, NY 10016, USA
fhu@gradcenter.cuny.edu

[2] Department of Computer Science, The City College of New York, New York, NY 10031, USA
zzhu@ccny.cuny.edu, jzhang@cs.ccny.cuny.edu

Abstract. In this paper, we propose a first-person localization and navigation system for helping blind and visually-impaired people navigate in indoor environments. The system consists of a mobile vision front end with a portable panoramic lens mounted on a smart phone, and a remote GPU-enabled server. Compact and effective omnidirectional video features are extracted and represented in the smart phone front end, and then transmitted to the server, where the features of an input image or a short video clip are used to search a database of an indoor environment via image-based indexing to find both the location and the orientation of the current view. 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 database indexing, and computation is accelerated by using multi-core CPUs and GPUs. Experiments on synthetic data and real data are carried out to demonstrate the capacity of the proposed system, with respect to real-time response and robustness.

Keywords: Panoramic vision · Mobile computing · Cloud computing · Blind navigation

1 Introduction

Localization and navigation in indoor environments such as school buildings, museums etc., is one of the critical tasks a visually-impaired person faces for living a convenient and normal social life. Despite a large amount of research have been carried out for robot navigation in robotic community[4][5][6], and several assistive systems are designed for blind people[15][11][1], efficient and effective portable solutions for visually impaired people are not yet available. In this paper, we intend to build an easy-to-use and robust localization and navigation system for visually-impaired people, using a portable omnidirectional lens on a mobile phone camera. Currently, the main stream solutions for localization are based on GPS signals; however, in an indoor environment these methods are not

L. Agapito et al. (Eds.): ECCV 2014 Workshops, Part III, LNCS 8927, pp. 600–614, 2015.
DOI: 10.1007/978-3-319-16199-0_42

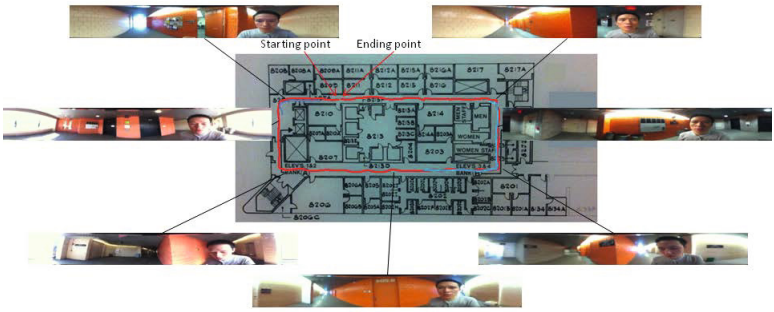


Fig. 1. One testing environment and some sample omnidirectional images. The red line in the map is the modeled path, and the dark blue line as well as the light blue line are the testing paths.

applicable since GPS signals are unavailable or inaccurate. Pose measurements using other onboard 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.

To provide an alternative solution to GPS-based navigation systems, RFID sensors and mobile robot based systems were developed by Kulyukin et al[9] and Cicirelli et al[3]. Although passive RFID tags can integrate local navigation measurements to achieve global navigation objectives, these systems rely heavily on the distribution of RFID sensors and specially designed robots. In our method, no extra sensors or infrastructure need to be installed in the environment, and no other complex devices are required except a daily-used smart phone (such as an iPhone) and a compact lens. Another existing solution proposed by Legge et al[10] uses a handheld sign reader and widely distributed digitally-encoded signs to give location information to the visually impaired. Again, this method also requires attaching new tags to the environment, and it can only recognize some specific locations. Our proposed system does not have any 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.

Our system has the following characteristics that make it an appropriate assistive technology for the visually impaired navigation in the indoor context. (1) No extra requirements are needed on hardware except a daily-used smart phone and a portable lens, which is simple, inexpensive and easy to operate. Neither extra power supply is needed. (2) A cloud computing solution is utilized. Only compact features of a video clip need to be processed in the front end and then be transmitted to a server, which guarantees a real-time solution while saving a lot of bandwidth. Different from transmitting an original image or a video clip, which will cost a lot of mobile traffic and may need a long communication time, our method only transfers essential scene features, usually less than one percent of the original data and thus has very low communication cost and

little transmitting time. (3) The system is scalable. Majority of localization and navigation algorithms are executed in the cloud server part, and image/video databases are stored in the cloud part too. This efficiently makes good use of the storage and computation power of the server and do not occupy too much of smart phone's resources. This also implies that the solution can scale up very well for a large database. (4) Parallelism in both data and tasks can be explored, since data parallelizing can be applied in both spatial and temporal dimensions of the video data, the localization algorithms can be accelerated by using many-core GPUs, and thus significantly reducing computational time. One example of our testing environments is shown in Fig. 1. The map in the middle is an indoor floor plan of a campus building. The red line in the map is the traversal path when modeling this environment, and the dark blue line and the light blue line are the paths used for capturing video clips for testing. Some sample omnidirectional images are shown around the map, and their geo-locations are indicated in the floor plan.

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 calibration and preprocessing procedure. Section 5 discusses localization by indexing as well as issues in paralleling process. Section 6 gives a conclusion and points out some possible future work.

2 Related Work

Appearance-based localization and navigation has been studied extensively in computer vision and robotics communities using a large variety of different methods and camera systems. Outdoor localization in the urban environment with panoramas captured by a multicamera system (with five side-view cameras mounted on the top of a vehicle) is proposed by Murillo et al[14]. Another appearance approach based on Simultaneous Localization and Mapping (SLAM) of a large scale road database, which is obtained by a car-mounted sensor array as well as GPS, is proposed by M. Cummins and P. Newman[4]. 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 by providing a robust and easy-to-use panoramic mobile navigation system.

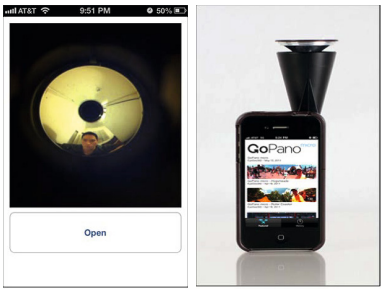
A visual nouns based orientation and navigation system for blind people was proposed by Molina et al[13][12], which aligns images captured by a regular camera into panoramas, and extracts three kinds of visual nouns features (signage, visual text, and visual-icons) to provide location and orientation 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

Fig. 2. System diagram

poor image resolution. In our paper, an omnidirectional lens GoPano[7] 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 is required, and no sign detection or recognition is needed.

Another related navigation method in indoor environments is proposed by Aly and Bouguet[2] 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 complete graph of every viewpoint to select triples for parameter estimations. In our method, since we use sequential video frames, we do not need to find such spatial relationships between images, and thus we can skip these procedures. Therefore we reduce the computation cost and pursue a real-time solution.

Representing and compressing omnidirectional images into compact rotational invariant features was proposed by Zhu et al[17], where the Fourier transform of the radial principle components of each omnidirectional image is used to represent the image. In our paper, based on the observation that an indoor environment usually includes a great number of vertical lines, we explore the omnidirectional features of these vertical lines in both the HSI space (Hue, Saturation and Intensity) and the HSI gradient space of an omnidirectional image, instead of using original RGB space only, as in [17], and generate one-dimensional omnidirectional HSI and HSI gradient projections. Then we use the Fourier transform components of these projections as the representation of omnidirectional image to reduce feature size and obtain rotation-invariant features and then find both the viewer's location and heading direction. Another major difference is that we aim to find a user's location and orientation from each omnidirectional image, where as in [17], only six road types are classified using a neural network based approach.



(a) (b)

Fig. 3. Smart phone GUI and omnidirectional lens

3 Overview of Our Approach

The hardware of the system is designed with two components: the smart phone front end part and the server part. The system diagram is shown in Fig. 2. The front end part consists of a smart phone and an omnidirectional lens, which mounts on the phone with a case. In our implementation, we use an iPhone and a GoPano lens, which is shown in Fig. 3.

The software of the system includes two stages: the modeling stage and the query stage. In the modeling stage of our current implementation, the developer of the database (the model) carries the mobile phone and moves along the corridors with an even pace, covering and recording the video into a database. Geo-tags(e.g. physical locations of current frames) are manually labeled for key frames (e.g. at doors, turns, etc.) and the rest of them are linearly interpolated. In the future, motion estimation algorithms can be used to ease the constraint of linear motion. To reduce the storage need, we do some preprocessing to the data, which will be discussed 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 wireless connections. The server receives the query and search the image candidates in the database, and returns the localization and orientation information to the user.

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 also not invariant to environment variables such as illumination, user heading orientation, and other environment factors. Therefore, we propose a concise and effective representation for the omnidirectional images by using a number of robust one-dimensional omnidirectional projections (omni-projections) for each panoramic scene. We have observed that an indoor environment often has plenty of vertical lines (door edges, pillar edges, notice boards, windows, etc.), and features along vertical lines can be embedded inside of the proposed omni-projection representations, so these features can be extracted and be used to estimate viewer's

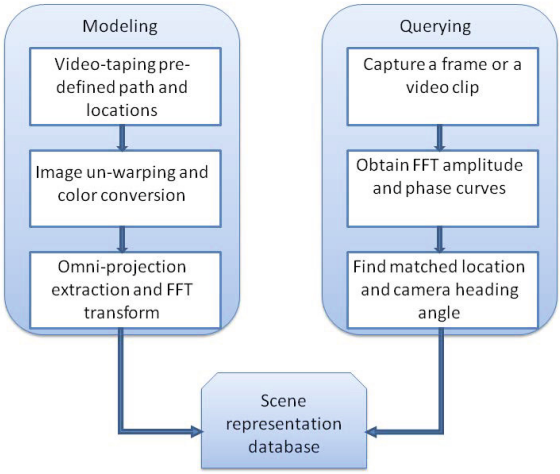


Fig. 4. Workflow of modeling and querying stages

locations. Newly extracted features of an input image are used as query keys to localize and navigate in the environment represented by the database.

Because different scenes may have similar omni-projection features, using a single frame may cause false indexing results. We 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 indexing. When 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 GPGPU to parallelize the query procedure and accelerate the querying speed.

3.1 The Overall Indexing Approach

While the system server is running, a user can send a query with newly captured video frames, and their corresponding locations are searched in the database. Since frames in the database are tagged with geo-location information, the system can provide the location information to the user. Currently, only physical locations relative to the floor plan are tagged, however in the future more information can be added, such as doorplates, office names, locations of daily-used facilities(e.g. telephones, water fountains, elevators) etc.

In the modeling stage, we first traverse all the desired paths and locations in an indoor environment and capture the original panoramic video of the scene. Then we perform un-warping to obtain the cylindrical representations of the omnidirectional images, convert the images to HSI color space and carry out a number of other preprocessing operations (such as gradient operations). After that, we project the images' columns to obtain the omni-projection curves for each frame (for both H, S, I and gradients of them). All the curves are normalized.

Finally we do the FFT transform to the curves and store the main components of the FFT amplitude curves and phase curves in the database.

In the querying stage, we first obtain normalized projection curves of the newly captured frames. Then we again carry out FFT transform to the curves and compute their FFT amplitude and phase curves. Using the amplitude curves, which are rotation-invariant, we search the frames in the database and find the closest matching frames. Using the phase curves of the omni-directional images, we can also estimate the relative rotation angle between a new frame and the matched frame in the database. The workflow of the modeling and querying stages in our system is shown in Fig. 4.

3.2 Cloud and Parallel Processing

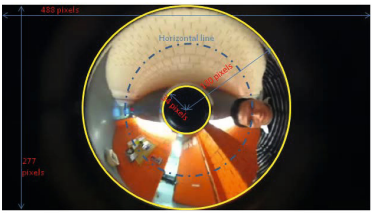
The modeling procedure is space intensive and the querying procedure could be very time consuming, so they will be manipulated in the cloud server part, where parallel processing can be applied to accelerate the procedures. The basic idea of multi-frames indexing is to use a sequence of newly captured video frames to query pre-built video frame database to increase the success rate. Even with the preprocessing step to reduce the size of the input data, the computational cost of multi-frame query in the database would be high if the database is large.

One strategy is to partition input video into individual frames and query the database with each frame. For every input query frame, a straightforward approach is to compare each frame with all the frames one-by-one. Then after all the queries return their matching candidates, which for example, top 5 matches for each frame, an aggregation step of the most consistent sequence of matches can be obtained by considering that both the input and the matched sequences are temporal sequences. In this way the querying process could have three levels of parallelisms. First, we process all the frames of a video in parallel. Second, we use multiple threads to compare each input frame with multiple database frames simultaneously, rather than comparing with them one-by-one. Third, 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 original omni-projection curves will be sent from the front end to the server, which is still efficient in communication due to their low dimensionality.

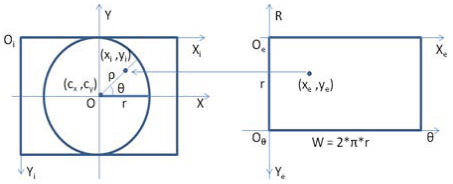
Parallel acceleration is necessary in the query procedure. Without parallel speed up, the time consumed in one single query operation take hundreds of milliseconds on our current experimental database (with only several thousand frames), and it would be multiplied if the scale significantly increases for a realistic indoor environment. After using parallel acceleration, this time is reduced to several milliseconds and greatly improves the performance of the query response. Details will be provided in Section 5.

4 Calibration and Preprocessing

The original frames captured by the GoPano lens and smart phone camera is a fish-eye-like distorted image, which is shown in Fig. 5(a), and has to be rectified.



(a)



(b)

(c)

Fig. 5. (a) Original video frame and its parameters; (b) geometric relationship between the original and un-warped images; and (c) un-warped omnidirectional image

Fig. 5(c) 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 this, a calibration procedure is needed to obtain all the required camera parameters.

Assuming that the camera is held up-right and the lens's optical axis is horizontal, the relationship between the original image and un-warped image can be illustrated in Fig.5(b)[8][16]. Define the original pixel coordinate system as $X_iO_iY_i$, the un-warped image coordinate system is $X_eO_eY_e$, and the original circular image center is (C_x, C_y). Then a pixel (x_e, y_e) in the un-warped image and corresponding pixel (x_i, y_i) in the original image has the following relationship.

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

where r is the radius of the outer circle of the original circular image, thus the vertical dimension (in pixels) of the cylindrical image, and $W = 2\pi r$ is the perimeter of the circle, turning to the horizontal dimension (in pixels) of the cylindrical image.

This un-warping process is applied to every frame in the database and all the input query frames. Since the 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. By doing this, the effective focal length in the vertical direction is also found. From this point on, we assume the image coordinates (u, v) are rectified from (x_e, y_e) , and the u direction (horizontal direction) represents the 360-degree panoramic view, and the v direction (vertical direction) is perspective.

We do not directly compare the pixel values of the frames; instead, we extract rotational invariant features from the FFT of HSI or HSI gradient curves, which are obtained by projecting the region of interest (ROI) of the image frame. Currently rectangular ROI is manually determined the same for all images in order to exclude most part of th floor regions that are subject to changes in different times. To represent a scene, six omni-projection curves are extracted from each rectified cylindrical image. Three of them are from H, S and I channels of the image, and the rest three are from the gradient magnitudes of the H, S and I channels. The gradient magnitude image of $f(u, v)$ can be calculated

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

where f can be H, S and I, $\frac{\delta f}{\delta u}$ is gradient in the u direction, and $\frac{\delta f}{\delta v}$ is gradient in the v direction. The magnitude of a gradient value is the L-2 norm of the $(\frac{\delta f}{\delta u}, \frac{\delta f}{\delta v})$. In practice, since we mainly focus on the vertical lines in the images as our features, we only use the horizontal gradient. From this point on, we will use a feature function g(u,v) to represent one of the six types of images (H, S, I and their gradient magnitudes, or simply gradients). Then 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 indices (corresponding angles from 0 to 360 degrees). Therefore the curve $c(u)$ is an omnidirectional projection curve (i.e., omni-projection curve). A linear normalization is applied to all the curves to turn them into the same scale. With all the six curves, we can store each and every of them into the database, and use them to compare with the new input curves to find the optimal location for a newly input frame.

5 Localization by Indexing

Localization is essential to visually-impaired people, since not only it provides the position and orientation of the user, but also it can supply additional information of the environment, e.g. locations of doors, positions of doorplates, etc.

When the database scale increases, using key features to do the indexing is necessary in order to obtain real-time performance without losing too much index accuracy. Meanwhile, sequential search is not practical when database scales is up significantly, therefore parallel searching is explored in this paper.

5.1 Indexing and Rotation Estimation

We use the major components of FFT transforms of the six omni-projection feature curves as the keys to do the indexing in this paper. For an 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 $\Phi = -2\pi u_0/N$[17]. 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 DFT of $c(u)$ is defined as a_k, the DFT of $c'(u)$ is defined as b_k, then

$$\begin{cases} a_k = \sum_{u=0}^{W-1} x(u) e^{-je2\pi ku/W}, k = 0, 1, ..., W-1 \\ b_k = a_k e^{-je2\pi ku_0/W}, k = 0, 1, ..., W-1 \end{cases} \tag{5}$$

The magnitudes of the omni-projection curves are rotation-invariant, i.e., $|a_k| = |b_k|$. Therefore we use the FFT magnitudes of the six omni-projection curves of a query frame to do indexing in the database. In our current experiments, the first half of the FFT magnitudes of each curve $|a_k|$ are used in the querying stage for indexing. The distances of the six curves between a query and each database frame are calculated, and a distance metric is chosen, e.g. using one of the most discriminative curves or use the average distance of the six, and the final match is the database frame that yields the smallest distance. From our experiments, we have found that the gradient of intensity curve provides the best discrimination.

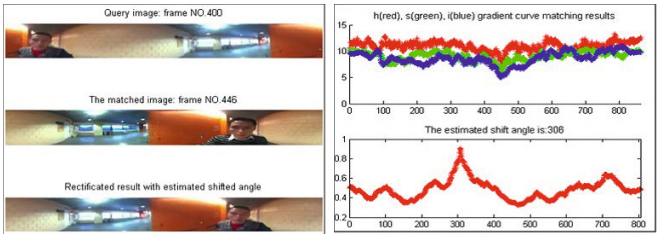
To find the rotation angle of the current image (i.e. amount of the circular shift), the problem is equivalent to finding the maximal value of the circular correlation function(CCF)

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

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

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

We only carry out this task after the optimal match is found, since it is meaningless to find the shifted angle if the location is not matched correctly.



(a) (b)

Fig. 6. (a) An example of a query image and its matched result in a database; (b) the matching scores with database frames and the estimated heading differences between the query and database frames

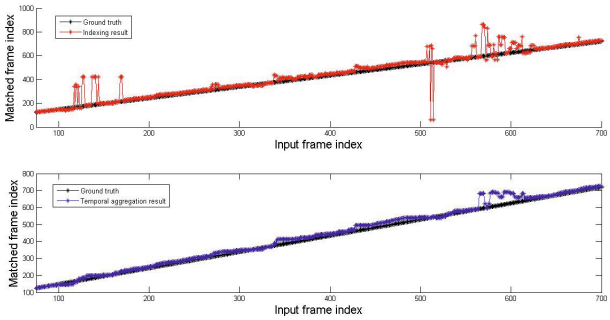


Fig. 7. Matching results of all the test database frames without (top) and with temporal aggregation(bottom)

5.2 Real Data Experiments: Accuracy and Time Performance

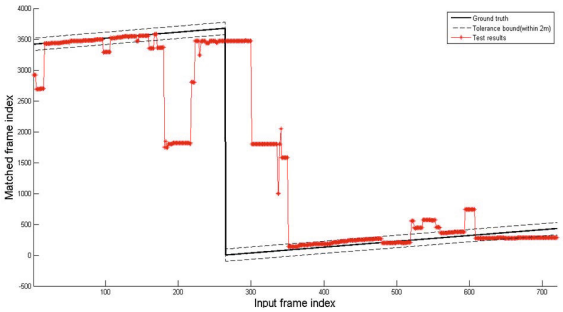
We carried ur a number of experiments for testing the accuracy and time performance of the indexing based localization approach. In the first experiment, we compared the accuracy of single versus multiple frame indexing with a small database of 862 frames. One example of indexing a new query frame is shown in Fig. 6 (a) and (b). Fig. 6(a) shows the input frame, the target frame, and the shifted version of the input image after finding the heading angle difference. In Fig.6(b), 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 on the bottom right, showing that the heading angle difference can be found after obtaining the correct 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 Fig. 7 (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

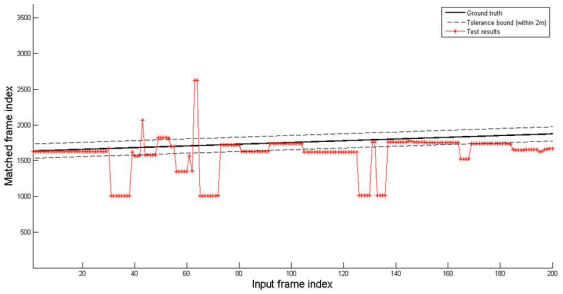
old sequence). Both sequences cover the same area. The black curve shows the ground truth data of matching results, and the red curve shows the matched results by our system. There are a few very obvious mismatches around frame 150, 500, and 600 due to the scene similarities. This leads us to design 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 results for all the input frames will yield a much more robust result. Fig. 7(bottom) shows the testing results with temporal aggregation, where for every frame, its nearby 25 frames' querying results are aggregated and the median index value is used as the final result. In the current implementation, 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 reduced to 14.7 frames with 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. But more importantly, as we can see from the curve, temporal aggregation has corrected the very obvious mismatches and generate more robust results.

In the second experiment,the entire floor of a building is modeled, as shown in Fig. 1. We first capture a video sequence of the entire floor (with a total of 3674 frames) as the training database, whose path is shown as the red line. We then capture two other short databases(along the dark blue line and the light blue line) as the testing databases. Some sample omnidirectional images used in the modeling process are shown around the map in Fig. 1, and their geo-locations are attached to the floor map too. Fig. 8 shows matching results using the two test databases against the large scale database. Fig. 8(a) shows the results using the frames near the starting and ending position. We would like to note that starting point and ending point are marked out only for visualization purpose; they are not used in the indexing process. Fig. 8(b) shows the results using frames in the middle of the loop. The black line is the ground truth, and the dashed black line is the tolerance bound with the accuracy within 2 meters. Because it is inevitable to avoid scene similarity, there are some mis-matching results. For example, in Fig. 8(c), the first image is the best matched result and the second one is the original query image, however they are images of two totally different locations. Again, these mismatches could be corrected using temporal aggregation.

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 use 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. Fig. 9 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



(a)



(b)

(c)

Fig. 8. (a) Matching results near the starting point of the loop; (b) matching results in the middle of the loop; (c) a pair of mismatching results

1000 to 8000 frames). Note that the current 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. 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.

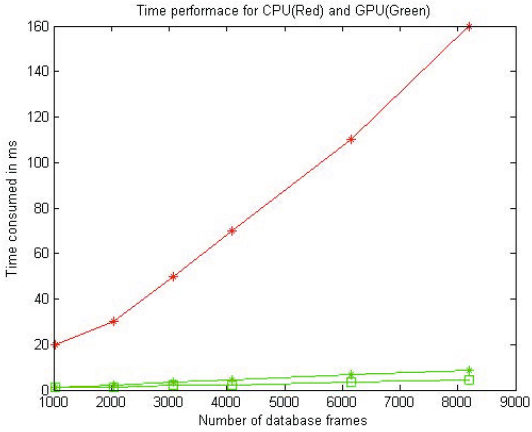


Fig. 9. Time usage with and without GPU acceleration: Red without GPU; Green with GPUs. Curves with squares are experiments using 1024 threads while curves with asterisks are experiments using 2048 threads.

6 Conclusion and Discussion

In this paper we have proposed a mobile panoramic system to help the visually impaired people to localize and navigation in indoor environments. We use a smart phone with panoramic camera and high performance server architecture to ensure the portability and mobility of the user part and take advantage of the huge storage as well as high computation power of the server part. An image indexing mechanism is used to find the rough location of an input image (or a short sequences of images), and to improve the query speed and ensure a real time performance, we use many-core GPUs to parallelize the query procedure. There are a few issues we will be dealing with in the future. First, a large scale scene database, for example, multiple floors of an entire building, or a number of buildings on campus, will be built and used to create more testing environments. Second, hierarchical and context-based methods can be used to avoiding sequential searching the entire database for every query. For example we can use GPS or WiFi to obtain rough location information and localize the user in the nearby places before we search the database around those locations. Third, user interface in communicating the localization and navigation information to a blind user, as well as implementation of the front end algorithms on the mobile phone should be optimized to make it more natural for the user to use.

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 the anonymous reviewers for their valuable comments and suggestions that have helped us revise the paper and identify future research directions.

References

1. Altwaijry, H., Moghimi, M., Belongie, S.: Recognizing locations with google glass: A case study. In: 2014 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 167–174, March 2014
2. Aly, M., Bouguet, J.Y.: Street view goes indoors: Automatic pose estimation from uncalibrated unordered spherical panoramas. In: 2012 IEEE Workshop on Applications of Computer Vision (WACV), pp. 1–8. IEEE (2012)
3. Cicirelli, G., Milella, A., Di Paola, D.: Rfid tag localization by using adaptive neuro-fuzzy inference for mobile robot applications. Industrial Robot: An International Journal **39**(4), 340–348 (2012)
4. Cummins, M., Newman, P.: Appearance-only slam at large scale with fab-map 2.0. The International Journal of Robotics Research **30**(9), 1100–1123 (2011)
5. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: Monoslam: Real-time single camera slam. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(6), 1052–1067 (2007)
6. Di Corato, F., Pollini, L., Innocenti, M., Indiveri, G.: An entropy-like approach to vision based autonomous navigation. In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 1640–1645. IEEE (2011)
7. GoPano: Gopano micro camera adapter, June 2014. http://www.gopano.com/products/gopano-micro
8. Hui, Z., Fei, L., Hui-juan, L.: Study on fisheye image correction based on cylinder model. Computer Applications **28**(10), 2664–2666 (2008)
9. Kulyukin, V., Gharpure, C., Nicholson, J., Pavithran, S.: Rfid in robot-assisted indoor navigation for the visually impaired. In: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems. (IROS 2004), vol. 2, pp. 1979–1984. IEEE (2004)
10. Legge, G.E., Beckmann, P.J., Tjan, B.S., Havey, G., Kramer, K., Rolkosky, D., Gage, R., Chen, M., Puchakayala, S., Rangarajan, A.: Indoor navigation by people with visual impairment using a digital sign system. PloS one **8**(10), e76783 (2013)
11. Manduchi, R., Coughlan, J.M.: The last meter: blind visual guidance to a target. In: Proceedings of the 32nd annual ACM conference on Human factors in computing systems, pp. 3113–3122. ACM (2014)
12. Molina, E., Zhu, Z.: Visual noun navigation framework for the blind. Journal of Assistive Technologies **7**(2), 118–130 (2013)
13. Molina, E., Zhu, Z., Tian, Y.: Visual nouns for indoor/outdoor navigation. In: Miesenberger, K., Karshmer, A., Penaz, P., Zagler, W. (eds.) ICCHP 2012, Part II. LNCS, vol. 7383, pp. 33–40. Springer, Heidelberg (2012)
14. Murillo, A.C., Singh, G., Kosecka, J., Guerrero, J.J.: Localization in urban environments using a panoramic gist descriptor. IEEE Transactions on Robotics **29**(1), 146–160 (2013)
15. Rivera-Rubio, J., Idrees, S., Alexiou, I., Hadjilucas, L., Bharath, A.A.: Mobile visual assistive apps: benchmarks of vision algorithm performance. In: Petrosino, A., Maddalena, L., Pala, P. (eds.) ICIAP 2013. LNCS, vol. 8158, pp. 30–40. Springer, Heidelberg (2013)
16. Scaramuzza, D., Martinelli, A., Siegwart, R.: A flexible technique for accurate omnidirectional camera calibration and structure from motion. In: IEEE International Conference on Computer Vision Systems, ICVS 2006, pp. 45–45. IEEE (2006)
17. Zhu, Z., Yang, S., Xu, G., Lin, X., Shi, D.: Fast road classification and orientation estimation using omni-view images and neural networks. IEEE Transactions on Image Processing **7**(8), 1182–1197 (1998)