Street navigation using visual information on mobile phones

Nguyen, Phuong Giang; Andersen, Hans Jørgen; Høilund, Carsten

Published in:
IEEE proceeding of the 10th International Conference on Intelligent Systems Design and Applications

DOI (link to publication from Publisher):
10.1109/ISDA.2010.5687295

Publication date:
2010

Document Version
Early version, also known as pre-print

Street navigation using visual information on mobile phones

Giang P. Nguyen Hans J. Andersen Carsten Høilund
Department of Architecture, Design and Media Technology
Aalborg University, Denmark
Email: {gnp,hja,ch}@create.aau.dk

***Abstract*—Applications with street navigation have been recently introduced on mobile phone devices. A major part of existing systems use integrated GPS as input for indicating the location. However, these systems often fail or make abrupt shifts in urban environment due to occlusion of satellites. Furthermore, they only give the position of a person and not the object of his attention, which is just as important for localization based services. In this paper we introduce a system using mobile phones built-in cameras for navigation and localization using visual information in accordance with the way we as humans navigate. The introduced method uses local features for extraction of natural feature points from images which are compared to a database for localization. The system is tested and evaluated in a real urban environment and the result shows very high success rate.**

***Keywords*-Visual-based navigation, mobile application.**

I. Introduction

Mobile phones have become very popular devices because of their reasonable prices, light weight, easy handling and rich set of functionality. Nowadays, people are not only using mobile phones for making calls or sending messages. The number of applications integrated on the phones is increasing along with more user interaction. These interactions are now not just limited between the user and his phone, but also extended to the user, the phone and the real world. In other words, the user uses his phone as an intermediate device to interact with the real world [8], [18], [5], [9], [2], [13]. For example, in [8], [13], where users interact with a poster through a mobile phone. Another example is in [18]. In this reference, a vision-based mobile application for in-store shopping is presented. That application helps users to get information about certain type of product while shopping.

Introducing location based services is one of the current trends within development of mobile phone applications. These developments are supported by system such as GoogleStreet [7] for geo-referencing of information. However, there is a limitation of GPS service subjectivity to occlusion of satellites and significant drifts. Instead, another approach is to use visual information of the surrounding environment, which contains a rich source of information for localization. This approach is also in accordance with the way we as human navigate and thus it may give a versatile and robust method easily adapted by the users. Example applications in this approach can be found in [3], [12], [17], [4], [6]. For instance, in [12], a mobile phone application is introduced to support pedestrian navigating in the street using panoramic landscape images. A mobile tour guide system is presented in [4], [6]. These systems provide users information of a certain point of interest such as buildings or statues that they are interested in.

The main contribution of the paper is that we develop a different scenario where we aim at providing the user information to navigate in unfamiliar areas. Assume that you have to go to a place and you don't know how to get there. In a traditional way, a paper map is used where you first have to locate your self and spend time to figure out the best route. In our application, the user only need to take a picture of a closed by building, and then gets all feedback regarding the location, and guiding instructions. An important part of such a system is the recognition process. When using buildings as visual information for localization, we have to take into account different factors that interfere the recognition performance. These factors include the changes of building appearance at different time of a day, different seasons due to light source, weather conditions, building decoration, and occlusion. Buildings can also be captured at different viewpoints, rotations and scales which is subjected to users. Figure 1 shows an example of different variations in the appearance of a building. These influences are main challenges to any recognition system. In the paper, we propose a system using local features to deal with these issues. Local features are extracted from areas centered at key points, which are corners in most of the cases. These features give better performance than traditional global features, such as color histogram, because of their stability under different imaging conditions [11], [16].

The paper is organized as follows. In the next section, we present computer vision techniques to recognize buildings. Then, we will describe in more detail how our scenario is set up and implemented. After that, user based evaluations and testing are described. Finally, our observations about the system and conclusions are discussed.

II. Building recognition using local features

In this section we describe how we implement building recognition. Our system contains three main steps. First, we select good features to represent image content. Second, extracted features are indexed and stored on a server. Finally, when a query image is sent to the server, the system searches through the database to find matching results.

Fig. 1. Example of a building under different variations.

A. Extracting local features

In order to get a good recognition system, the main issue is having a reliable method that can detect locations using natural images under changing weather condition, time of day and season and significant variation of geometrical condition including scale, orientation, and viewpoint. In other words, selected image features should be invariant, i.e. have minimal changes, under these imaging conditions. Scale invariant image features (SIFT) is a popular approach for extracting local features in image recognition [10], [16], [11]. In these references, SIFT is demonstrated among the best local features. In [15], we have shown that a more recently developed feature called multi-scale oriented patches (MOPS) [1] gives a better performance than SIFT. Therefore, in our application, we apply this feature to recognize building. MOPS can be briefly described in three main steps:

- An input image is incrementally smoothed with a Gaussian kernel. From there an image pyramid is constructed by down-sampling each smoothed image.
- In each image from the pyramid, key points are extracted using Harris corner detector.
- Descriptors are computed from a window of size 28×28 drawn centered at each candidate key point. The descriptors contain orientation of the window, the scale level where the key point is found, and a 64 dimensional feature vector of gray-scale histogram.

B. Indexing local features

All extracted features will form a 64 dimensional feature space. A common problem using local features is having a large number of features in each image. To overcome this, an indexing technique for fast searching through the feature space is employed. In [14], the authors introduce a vocabulary tree for indexing and searching through a large data. A vocabulary tree is a tree that branches starting from centers of data. It is defined as an hierarchically quantization built by hierarchical k-means clustering in [14]. A vocabulary tree is recursively formed in the following two steps:

- K-means is performed on the selected data.
- Data is quantized into k clusters according to the closest centroid.

This hierarchical k-means is controlled with two parameters, namely levels l and clusters k. The former determines the number of recursive division of data, and the later determines how many clusters to create within each division. First, a virtual root is first defined. We then cluster all data points into k groups, and define k cluster centers. Each cluster is then partitioned again into k groups, where each group consists of points which have their feature vectors closest to the centroid. The process is repeated until it reaches the number of level l. We use the same values set as in [14] for these two parameters i.e. $l = 6$ and $k = 10$. This results in 1 million leaf nodes on the tree.

C. Building recognition system

At the server side, after MOPS features are extracted from images as described in section II-A, a vocabulary tree is constructed to store all the features. There is an inverted file list located at each leaf nodes of the tree. This file contains all images that are listed if they have one or more features routed through a corresponding branch.

In the searching and matching step, features extracted from a query image are then propagated down the tree until reached to the leaf ends. At each level of the tree, feature vectors are compared to the cluster centers and the closest cluster is found. The path down the tree is encoded by a single integer and then used later in scoring. The scoring process is able to give a ranked list of images matched with the query one. After the scores are calculated at each leaf, they are added to all images in the inverted file list of that branch. The image with highest score is assumed to be the best match.

III. Visual-based navigation

A. Scenario setup

In this section, we describe our mobile phone application for users to navigate on the street. We created a scenario where users were placed in unfamiliar areas. Each user was given the task to go from one location to another. The user was provided with a mobile phone with a built-in camera. While navigating on the street, and in need of route instruction, the user captured a picture of a nearby building and sent the picture to the server to ask for further guidance.

At the server, the system processes the image and searches through the database to find possible matches. Matching images and associated information are sent back to the user. The process is illustrated in 2. Our test area was carried out at the campus of Aalborg University. In this area, we chose 20 buildings for testing the recognition system. We took pictures of those buildings at different times of day and on different days. The dates were ranged from winter to summer time. In total, we collected a dataset of 435 images of the chosen 20 buildings and some other buildings in the test area. The reason for taking pictures of other buildings besides the 20 ones is to create larger variety in the database so that it is more difficult for the recognition system. All tests used HTC

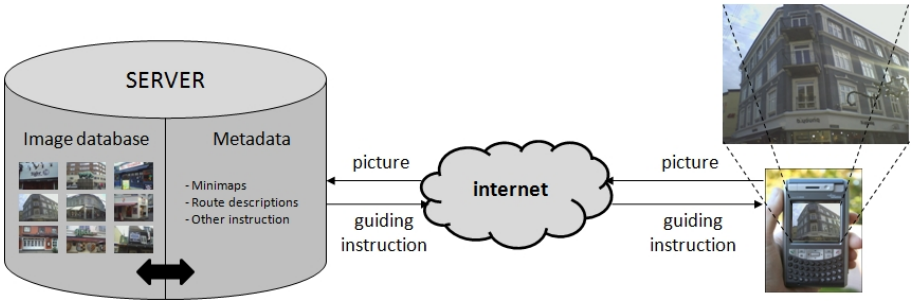

Fig. 2. An overview of the client-server system for visual-based navigation on streets.

Touch Diamond2 mobile phones with built-in cameras and the recognition process was run on a MacBookPro with an Intel Core Duo T2600 processor at 2.16GHz. Communication between mobile phones and the laptop was via a 3G connection.

B. Application design

At the starting point, a welcome screen with different destinations was shown to the user. By selecting one image, he chose the destination and started the application. Figure 3 shows an interface from the mobile phone of this step, information of the selected destination was displayed at the bottom of the screen. Then, an overview map with a route drawn from the starting point to the destination was shown to the user. The purpose of this interface is to give the user an overall view of the journey as well as estimated walking time.

Along the route, we marked a number of buildings as waypoints. Depending on how far away the destination is, a number of way-points were defined. This number was determined beforehand and stored for each route. In our first test, the distance between two way-points was approximately 300 meters. For the current design of the system, this value is experimentally chosen such that an user can easily remember the route within such distance. For the final system, this value can be flexible and automatically adjusted depending on the type of a route. For example, if it is a straight route, then a longer distance can be used, while sorter route is applied if the route is more complicated. Another criteria can be used to tune this value, which is related to users such as how well the user processes the map and he/she can decide to adjust the distance himself/herself.

After deciding the distance between two way-points, the overview route was divided into several sub-routes. Therefore, the user worked only on one sub-route at a time. This makes it easier for the user to remember the map, especially when the destination is too far away. Figure 5 shows an interface with the sub-route map, which we call a minimap. This interface shows the route from the current location to the nearest way-point with a route description. We also displayed a picture of the next way-point, so that the user knows what he should be looking for. When the user reaches the next way-point, he will click on the button ”I am here” to get further instructions.

To make sure that the user actually got to the right waypoint, we asked him to take a picture of the building and send to the server. The server performed the building recognition process to find a list of matching images and returned a list of top 5 best matched. The user chose one image that he thinks is the most similar to the photo he had taken (figure 6). In case the system failed at finding at least one correct match the user was suggested to take another picture. This is because this failure can be caused by different factors other than the recognition system itself, such as the quality of the query image being too blurry or out of focus. If the user can find an image from

Fig. 3. Welcome screen

Fig. 4. Overview route

Fig. 5. Sub-route

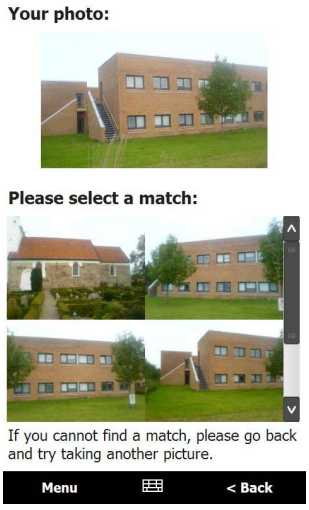

Fig. 6. Retrieved information

the matching list, a new sub-route interface was shown with new minimap and route description. These steps were repeated until the user reached his destination.

C. Evaluation

1) Recognition performance: Our first experiment is to evaluate the performance of the recognition system as this part plays the most important role in the successful of the application. In order to test the system, we took different pictures of same buildings in the database. This means that for each of the 20 buildings, we took a set of images as a query set. To simulate different possibilities of users taking pictures, query images are taken at different orientations, viewpoints, and scales. Our query set contains 36 images. These images are then sequentially sent to the server for matching.

The recognition rate is computed by calculating how many correct matches are returned in the top 5 result. To do so, each query is manually labeled its ground-truth i.e. corresponding building category, and if a returned image belongs to the same building then we count it as a correct match and vice versa. The recognition rate is averaged over 36 query images. Because the user is allowed to select one correct from the list of returned images, we also compute the chance of having at least one correct match in the returned list, which is called the estimated success. Therefore, we have:

$$\begin{aligned} \text{Recognition rate} &= \frac{\sharp \text{ of correct matches}}{\sharp \text{ of returned images}} \\ \text{Estimated success} &= \min(\sharp \text{ of correct matches}, 1) \end{aligned}$$

Result shows that if more than 2 images are returned, we have an approximately 90% chance of receiving a correct image. This number is rather high when considering the large varieties of image appearance due to user's freedom of taking input pictures.

2) Application performance: After evaluating the system performance, we invited 13 users to participate and test the application. The selected participants were not familiar with the application as well as the test area. We set up four different routes from one location to another in the Aalborg University campus area. The four destinations were selected from 20 buildings. Each route is contained at least 3 way-points, i.e. test users had to pass through those way-points to reach the destination. For example, figure 7 shows an example of a route from building 1 to building 14 which passes through 3 waypoints, namely building 18, 13, and 15.

For an even selection of destinations, we selected in advance the destination for each user. This means that we had at least 3 users for each route. Given the destination, the user used the phone to navigate following the instruction from the application. At the starting point, the application showed a minimap with an example image of the building that the user should look for. The minimap was showed with the route that the user should follow. Route description was provided for clearer information. When the user found a building that is similar to the example, he was guided to take a picture of that building. When he got the feedback from the application with a list of suggested similar images, he chose one from them and continued the journey to the next building. In case the application did not return any similar images, the user was asked to try taking new picture. We did not restrict the number of tries from users, so it was depended on whether they wanted to continue or give up. If the user reached the given destination, we reported it as a success case. Otherwise, we reported a failure case. User actions were recorded at each building during the test, with 3-4 test users for each route.

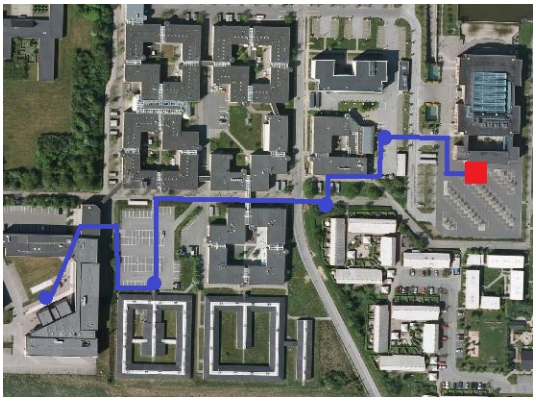

Fig. 7. An example of a route passing through 3 way-points.

The results are summarized in table I. The first column is the user ID, from 1-13. The last column shows whether the user succeeded or not (Yes/No). Each of the remaining columns is the number of times that a user has to take a picture to find a similar image, at each building. For example, in route 1, users had to go to five buildings including the destination building, namely building 2, 3, 12, 14 and 15. At each of these buildings, they took pictures until they got a similar one from the matching list. The ideal case would be only one shot per building. User 1 got through all first four buildings with his first try, but at the last building (his destination) he had to take 8 shots but still did not succeed. User 6 got a success route with first try in most of buildings except building 3, requiring two attempts.

We notice that failure cases are mainly caused by the sunlight going straight into the camera when the users took pictures (see figure 8(b) for an example). Therefore, the recognition failed at retrieving correct images. This can also be observed from the summary table that certain buildings required more attempts to pass through. The other reason is the different ways of taking pictures by users. Some users stood from long distance, and others tried to get close to buildings to take pictures.

In figure 8, we show some examples of different pictures of the same buildings taken by test users. Examples in figure 8(a) and 8(b) are buildings that often required more than one shot. In these pictures, there was a lot of sunlight that makes the appearance of the building changed. Figure 8(c) shows an example where a user accidentally covered the camera while taking a picture. These examples illustrate that the users had a freedom in taking picture. The same building can be taken at different viewpoints, rotations, and scales. Therefore, there was a big challenge for our application. However, we obtained a success rate of 84%. This is a very promising result for

UserID	Building 02	Building 03	Building 12	Building 14	Building 15	Success route
1	1	1	1	1	8	No
6	1	2	1	1	1	Yes
12	1	1	1	1	3	Yes

UserID	Building 02	Building 03	Building 10	Building 11	Success route
2	1	1	1	5	Yes
9	1	1	1	4	Yes
10	1	1	1	3	Yes

UserID	Building 02	Building 03	Building 17	Building 12	Building 19	Success route
3	1	1	2	1	5	Yes
7	1	1	1	1	6	No
13	1	1	1	1	4	Yes

UserID	Building 05	Building 04	Building 06	Building 17	Success route
4	1	1	1	1	Yes
5	2	1	1	1	Yes
8	3	1	3	1	Yes
11	2	1	1	3	Yes

TABLE I
REPORTING USER ACTIONS FOR ROUTE 1,2,3, AND 4. AT EACH TABLE, THE FIRST COLUMN IS AN ID OF A TEST PERSON. THE LAST COLUMN INDICATES WHETHER OR NOT THE USER SUCCESSFULLY REACHES THE DESTINATION. REMAINING COLUMNS SHOW THE NUMBER OF CAMERA SHOTS THAT THE USER TOOK UNTIL HE/SHE FOUND A CORRECT MATCH.

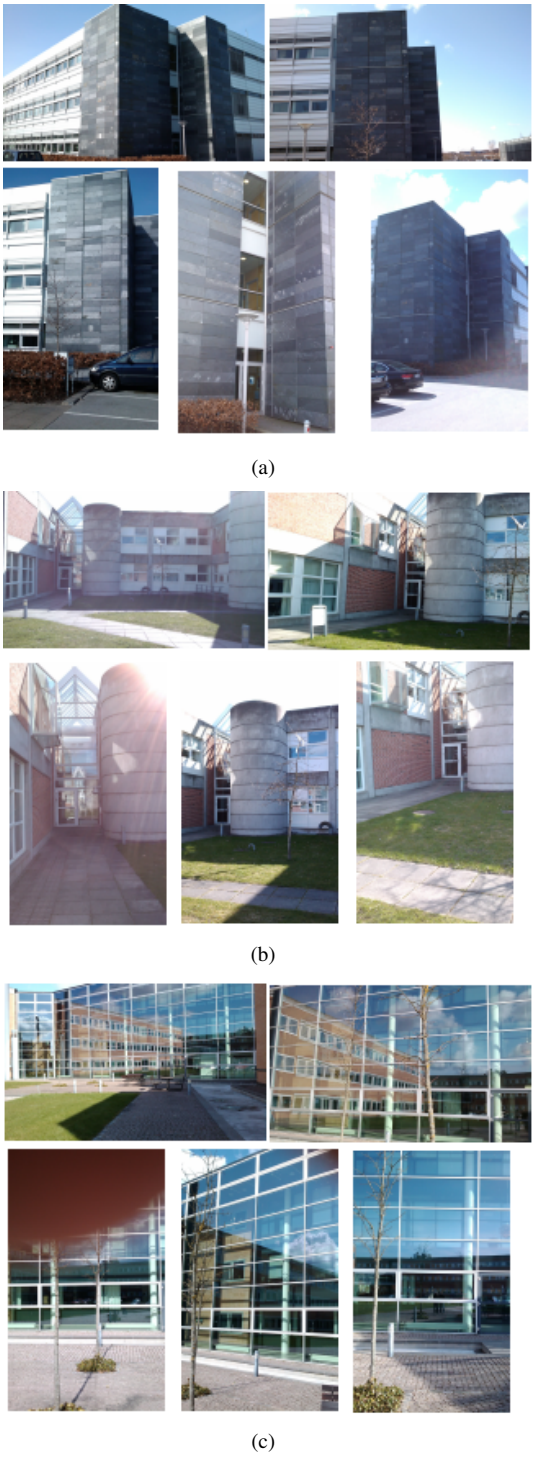

Fig. 8. Some examples of variety in user taking pictures of the same building. As users provided no restriction in the way they should take a picture, the same building was taken at different viewpoints, scales, and orientations. Light sources such as bright sunny day, shadow, reflection also added more challenge to the system.

further development of the application.

At the end of the test, users were asked to answer a questionnaire. Our questionnaire was divided into two parts. First was the performance, where we want to get users opinion on the recognition system and the total waiting time to get feedback from the application with the list of similar images. The second part was whether the application is easy to understand and use. Users answers are ranked from 1: strongly agree, 2: agree, 3: neutral, 4: disagree, and 5: strongly disagree.

For answering the question about system performance "I find the system was able to find similar images", there are 15% users gave strongly agree, 70% users with agree, and 15% were neutral. Regarding "the waiting time to get the list of similar images", 85% thought this was acceptable. 93% of the test users stated that the application was easy to understand and use. For evaluation on "the overall performance of the application", 77% agreed that they satisfied with the application, and 23% gave neutral answers. We also asked extra question "if the users prefer using this application over a paper map", to which we received very broad answers. While some users personally like using paper maps, the others prefer using new technology. We have 40% preferred using the application, 30% on neutral answers and 30% preferred the paper map.

We also recorded the processing time for all pictures taken. As a real-time application, time is also an essential factor. We recorded the recognition processing time, which is the time that is needed to complete the recognition process. The waiting time means the total time the user has to wait after sending a query image to the server, and get the information back. This means that after the user takes a picture of a building, a timer starts when he presses the "Get information" button, and

finishes when all information is received from the server and displayed to the user. Figure 9 shows the total response time as a bar plot. The left-most bar shows the time it takes to upload the image to the server, while the middle bare shows the time it takes to receive a reply from the server. The right-most bar is the time it takes to display the result to the user, i.e. the internal processing on the phone. The three bars are stacked to indicate the total time, which is what the user experiences. For all 13 users, we collect 100 pictures, in total.

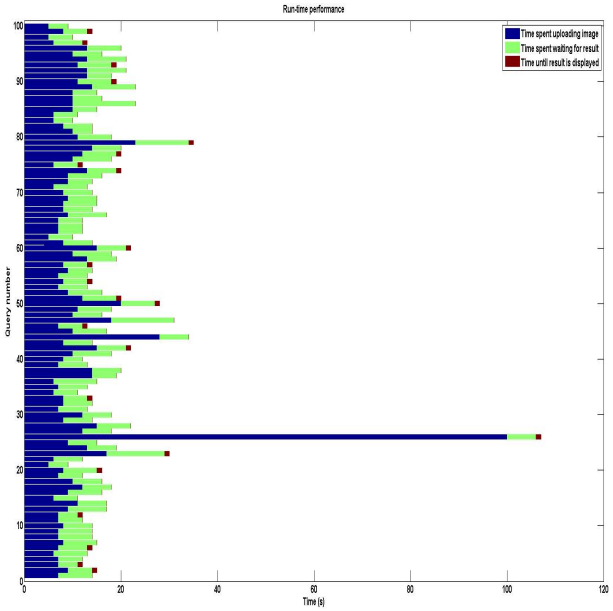

Fig. 9. Run-time performance. There was a long delay at query 26 because of the 3G connection issue at that certain moment.

It is observed from the figure that most of the time is spent uploading the image. The upload time is strongly dependent on the 3G connection in the test area. We noticed that at query 26, the upload time was very high due to an unstable connection at that time. To get the average time, we exclude query number 26. As a result, the upload time is 9.67 seconds, the recognition processing time is 6.28 seconds, and the display time is 0.22 seconds. The total time from the user presses "check match" to the result is displayed on the screen is therefore on average 16.17 seconds.

In a larger scale of database, a filtering step can be applied before the matching starts. Despite poor performance of GPS device, an approximated area using GPS can be used to filter the searching instead of an exhausted search in the database.

IV. Conclusion

With the development of technology, mobile devices have become very popular with many useful applications. In this paper, we present an application on mobile phone using image recognition techniques. Our application aims at supporting users in navigating through streets to get to their destination. The application provides feedback with map and guided directions when the user sends a picture of a nearby building. Using visual information of surrounding environment for navigation is natural for humans.

Our tests with users in a real urban environment demonstrate a high suitability of the proposed application in real-life situations. Future research is to provide more freedom to users in the sense that they free to choose which building to take a picture of. This means that, instead of fixed way-points, any nearby building can be used as query. This requires a huge collection of images at the server, which is similar to [7], where every building has to be captured and analyzed. This will also open a new challenge in storage and searching strategy with very large databases.

References

[1] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 510–517, 2005.

[2] E. Bruns and O. Bimber. Adaptive training of video sets for image recognition on mobile phones. *Personal and Ubiquitous Computing*, 13(2):165–178, 2009.

[3] T. Chen, K. Wu, K. Yap, Z. Li, and F. Tsai. A survey on mobile landmark recognition for information retrieval. In *Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pages 625–630, 2009.

[4] N. Davies, K. Cheverst, A. Dix, and A. Hesse. Understanding the role of image recognition in mobile tour guide. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices and services*, pages 191–198, 2005.

[5] P. Foeckler, T. Zeidler, B. Brombach, E. Bruns, and O. Bimber. Phoneguide: Museum guidance supported by on-device object recognition on mobile phones. In *Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, pages 3–10, 2005.

[6] G. Fritz, C. Seifert, L. Paletta, P. Luley, and A. Almer. A mobile vision service for multimedia tourist applications in urban environment. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2006.

[7] Google. Street view: explore the world at street level, 2010.

[8] N. Henze, T. Schinke, and S. Boll. What is that? object recognition from natural features on mobile phone. In *Proceedings of Mobile Interaction with the Real World*, 2009.

[9] M. Jia, X. Fan, X. Xie, M. Li, and W. Ma. Photo-to-serach: using camera phones to inquire of the surrounding world. In *Proceedings of the 7th International Conference on Mobile Data Management*, page 46, 2006.

[10] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[11] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[12] Y. Miyazaki and T. Kamiya. Pededtrian navigation system for mobile phones using panaramic landscape images. In *Proceedings of the International Symposium on Applications on Internet*, pages 102–108, 2006.

[13] A. Morrison, A. Oulasvirta, P. Peltonen, S. Lemmela, G. Jacucci, G. Reitmayr, J. Nasanen, and A. Juustila. Like bees around the hive: A comparative study of a mobile augmented reality map. In *Proceedings of the 27th international conference on Human factors in computing systems (CHI)*, pages 1889–1898, 2009.

[14] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006.

[15] The reference is removed for anonymous submission.

[16] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[17] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134, 2008.

[18] Y. Xu, M. Spasojevic, J. Gao, and M. Jacob. Designing a vision-based mobile interface for in-store shopping. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*, pages 393–402, 2008.