An Augmented Reality Tourist Guide on Your Mobile Devices

Maha El Choubassi, Oscar Nestares, Yi Wu, Igor Kozintsev, and Horst Haussecker

Intel Corporation
2200 Mission College Blvd
Santa Clara, CA 95052, USA
{maha.el.choubassi,oscar.nestares,yi.y.wu,igor.v.kozintsev, horst.haussecker}@intel.com

Abstract. We present an augmented reality tourist guide on mobile devices. Many of latest mobile devices contain cameras, location, orientation and motion sensors. We demonstrate how these devices can be used to bring tourism information to users in a much more immersive manner than traditional text or maps. Our system uses a combination of camera, location and orientation sensors to augment live camera view on a device with the available information about the objects in the view. The augmenting information is obtained by matching a camera image to images in a database on a server that have geotags in the vicinity of the user location. We use a subset of geotagged English Wikipedia pages as the main source of images and augmenting text information. At the time of publication our database contained 50 K pages with more than 150 K images linked to them. A combination of motion estimation algorithms and orientation sensors is used to track objects of interest in the live camera view and place augmented information on top of them.

Keywords: Mobile augmented reality, image matching, SIFT, SURF, location and orientation sensors, optical flow, geotagging.

1 Introduction

In the past few years, various methods have been suggested to present augmented content to users through mobile devices [1,2,3,4,5]. Many of the latest mobile internet devices (MIDs) feature consumer-grade cameras, WAN and WLAN network connectivity, location sensors (such as Global Position System - GPS) and various orientation and motion sensors. Recently, several applications like Wikitude (`www.wikitude.org`) for G1 phone and similar applications for iPhone have been announced. Though similar in nature to our proposed system, these solutions rely solely on the location and orientation sensors, and, therefore, require a detailed location information about points of interest to be able to correctly identify visible objects. Our system extends this approach by using the image matching techniques both for recognition of objects and for precise placement of augmenting information.

In this paper, we demonstrate *a complete end-to-end* mobile augmented reality (MAR) system that consists of Intel® Atom^{TM} processor-powered MIDs and Web-based MAR service hosted on a server. On the server, we store a large database of images crawled from geotagged English Wikipedia pages that we update on a regular basis. The MAR client application is running on a MID. Figure 1 demonstrates a snapshot of the actual client interface. In this example the user has taken a picture of a Golden Gate bridge in San Francisco. The MAR system uses the location of the user along with the camera image to return top 5 candidate matching images from the database on the server. The user has an option of selecting the image of interest, which retrieves a corresponding Wikipedia page with an article about the Golden Gate bridge. A transparent logo is then added to the live camera view "pinned" on top of the object (Figure 1). The user will see the tag whenever the object is in the camera view and can click on it later on to return to the retrieved information.

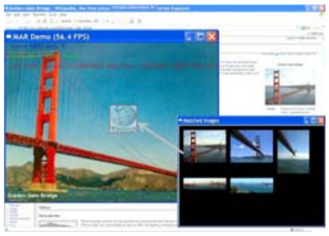


Fig. 1. A snapshot of our MAR system

The following are the main contributions of this paper:

- In [6], we relied on location and orientation sensors to track objects. In this paper, we additionally use the image content for tracking objects using motion estimation, thus improving the precision of placement of augmenting information in the live camera view.
- We created a representative database sampled from the large Wikipedia database to perform systematic testing of the MAR performance in a realistic setting.
- We propose a new method to improve the matching algorithm based on histograms of minimum distances between feature descriptors and present the results for our sample database.
- We propose a simple and promising way to extend our database of images by combining text and GPS information and various sources of images like Google image search and Wikiepdia. As a result we enhance the accuracy of image matching of our system.
- We implemented a fully-functional client application and optimized its performance for Intel® Atom^{TM} processor.

The organization of our paper is as follows. In Section 2, we give an overview of our MAR system. In Section 3, we describe the small sample Wikipedia database.

In Section 4, we present an enhanced version of the matching algorithm based on our testing results on this database. In Section 5, we extend our matching algorithm by investigating histograms of minimum distances between descriptors. In Section 6, we supplement our database by images from Google image search and show better image matching results on our sample database. In Section 7, we illustrate our algorithms with experimental results. In Section 8, we explain some of the implementation and code optimization aspects. Finally, we conclude in Section 9.

2 System Overview

In this section, we present the overview of MAR system as illustrated in Figure 2. The system is partitioned into two components: client MID and a server. The MID continuously communicates with the server through WAN or WLAN network.

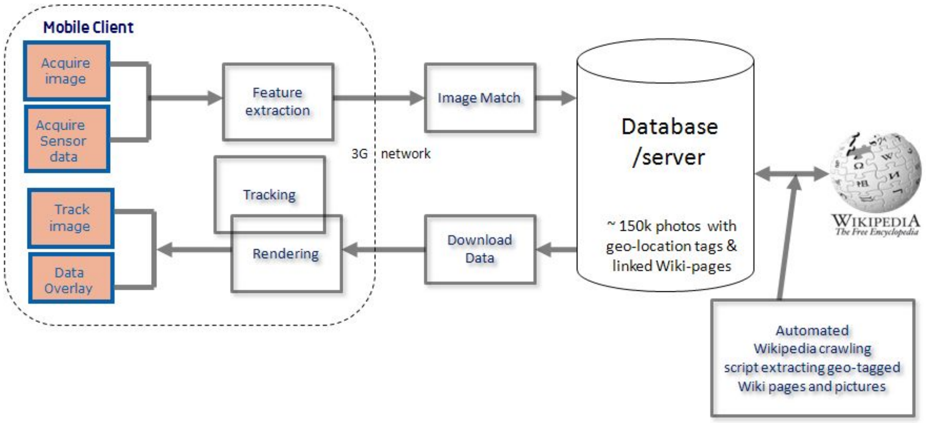

Mobile Client
Acquire image
Acquire Sensor data
Track image
Data Overlay
Feature extraction
Tracking
Rendering
3G network
Image Match
Download Data
Database /server
~ 150k photos with geo-location tags & linked Wiki-pages
WIKIPEDIA The Free Encyclopedia
Automated Wikipedia crawling script extracting geo-tagged Wiki pages and pictures

Fig. 2. MAR system Diagram with the image acquisition, feature extraction, rendering, and tracking on the client side, and the database and image matching on the server side

2.1 Client

– **Query acquisition.** Client MID devices contain cameras, orientation and location (such as GPS) sensors. Client continuously acquires live video from the camera and waits for the user to select the picture of interest. As soon as a picture is taken, sensor data from location and orientation sensors is written to the images EXIF fields.
– **Feature extraction.** Visual features are then extracted from the picture. We use 64-dimensional SURF [7] which are fast to compute compared to SIFT [8]. Client sends extracted feature vectors and recorded sensor data to

the server for searching matched images and related information. The reason of sending the compact visual features instead of full resolution images is to reduce network bandwidth, hence reduce latency.

– **Rendering and overlay.** Once the matched information is found, related data including matched image thumbnail, wikipage link, etc. will be downloaded to the client via WAN or WLAN network. Client will render the augmented information such as wiki tag related to the query object and overlay it on the live video. Our device has orientation sensors: a compass, an accelerometer, and a gyroscope. As the user moves the device around, the augmented information representing the query will be pinned to the position of the object. This way, multiple queries can be interacted with separately by simply pointing the camera at different locations.
– **Tracking.** When a query is made, the direction of the MID is recorded using the orientation sensors on the device. Client will continuously track the movement of orientation sensor [6]. However, tracking using orientation sensors is not very precise. We extend our tracking method to also include the visual information. The image based stabilization is based on aligning neighbor frames in the input image sequence using a low parametric motion model. The motion estimation algorithm is based on a multi-resolution, iterative gradient based strategy [9], optionally robust in a statistical sense [10]. Two different motion models have been considered, pure translation (2 parameters) and pure camera rotation (3 parameters).

2.2 Server

– **Database collection.** We crawled Wikipedia pages, in particular those with GPS information associated with them [6]. We downloaded the images from these pages to our server, extracted visual features from images and built our image database. At the time of publishing [6], the database had 50K images. Currently, it has over 150K images and it is constantly growing. This fact further emphasizes the need and the importance of applications exploring large resources of images.
– **Image match.** Once the server receives query from client MID, the server will perform image match. For image matching, we combine both the GPS information and the image content. More explicitly, we restrict the database of candidate matching images to those within certain search radius from the user GPS location. Next, we use computer vision techniques to match the query image to the GPS-constrained set of images. Downscaling the database from 150K images to a much smaller candidate image set after GPS filtering enhances the performance of image-based matching algorithms. If the number of nearby candidates is reasonable (< 300) we perform brute-force image matching based on the ratio of the distance between the nearest and second nearest neighbor descriptors [8]. Otherwise, for scenarios with highly dense nearby images or with no GPS location available, the database is large and we use indexing as proposed by Nister & Stewenius in [11]. Details of the matching algorithm are presented in the following sections.

3 Building a More Realistic Database

In our previous paper [6], we showed quantitative results on the standard ZuBuD test set [12] with 1005 database images and 115 query images. Also, we only had qualitative results for a small benchmark of images of iconic landmarks in an effort to represent MAR. Although the results on the ZuBuD set are informative, they do not precisely reflect the performance of the actual MAR system. To better represent MAR, we select 10 landmarks around the world: *the triumph arc* (France), *the Pisa tower* (Italy), the *Petronas towers* (Malaysia), *the colosseum* (Italy), *the national palace museum* (Taiwan), *the golden gate bridge* (California, USA), *the Chiang Kai-Shek memorial hall* (Taiwan), *the capitol* (Washington D.C., USA), *the palace of justice* (France), and *the Brandenburg gate* (Germany). For each landmark, we follow the steps of the MAR system. First, we filter the 150K images database on the server to a smaller set that includes only the images within a radius of 10 miles[1]. Next, we randomly pick one of the landmark images in the set as a query image, and keep the others as the database images. We now have a set of 10 query images and 10 databases. The total number of images in all the databases is 777. Please refer to Figure 3 for samples from these databases. Each row corresponds to one landmark and the leftmost image is the chosen query image. In the leftmost column, we show the number of relevant images out of the total number of images in the database. For example for the triumph arc landmark, the database has 5 images of the arc out of all the 27 images inside it. Moreover, the characteristics of these databases vary. For instance, the size of the database is only 5 for *the national palace museum*, while it is 200 for the justice palace. Also, the database corresponding to the golden gate bridge has 12 relevant images out of 19 images, while that of *the justice palace* has only 1 relevant image among 200 images. Finally, we manually annotate the databases. Simulating MAR on these databases, we obtain representative quantitative results for its performance since such databases are sampled from the Wikipedia data on our server, i.e, the actual database queried by MAR. Note that images labeled by the same landmark might look different depending on the lighting conditions, the viewpoint, and the amount of clutter.

4 Matching Enhancement by Duplicates Removal

4.1 Original Matching Algorithm

In order to recognize the top matching database images, our algorithm inspects each database image at a time and compares it to the query image. More specifically, for each SURF keypoint in the query image, the algorithm finds the nearest and the second nearest neighbor keypoints in the database image based on the L_1 distance between descriptors. Next, it computes the distances ratio and decides whether the query keypoint matches the nearest keypoint in the image database as in [8]. After the algorithm detects the matching pairs between the

[1] Our MAR system uses this radius also.

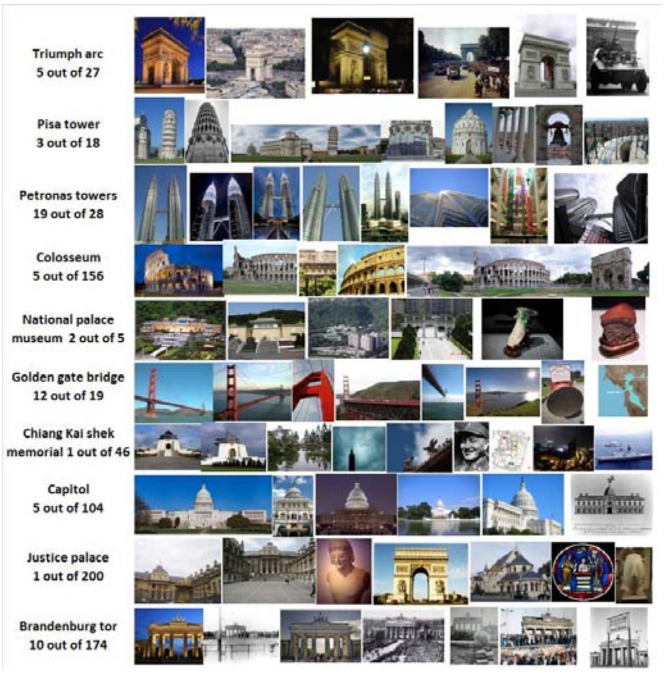


Fig. 3. Sample images from the GPS-constrained databases for 10 landmarks

query image and all the database images, it ranks these images in a descending order according to the number of such pairs.

4.2 Duplicates Removal

Running MAR on the data collected in Section 3, we identified one problem in the matching algorithm that degrades MAR's performance: having multiple matching keypoints in the query image corresponding to one keypoint in the database image magnifies false matches and degrades the matching accuracy. We adjusted the algorithm to remove duplicate matches and improve matching accuracy at no computational cost as shown below.

Example. In Figure 4, we see the top 5 images returned by MAR for the golden gate bridge query. The features used are SURF with a threshold[2] of 500. Clearly, the top 1 image is a mismatch.

In Figure 5, we see that many of the matches between the query image and the top 1 image are actually duplicates: for example, the 14 keypoints in the query image correspond to only one keypoint in the database image. Moreover, these matches are actually false. It turns out that many of the 60 matches are due

[2] The threshold is a parameter in the SURF algorithm. The larger this parameter is, the less sensitive is the keypoints detector and the smaller is the number of keypoints.

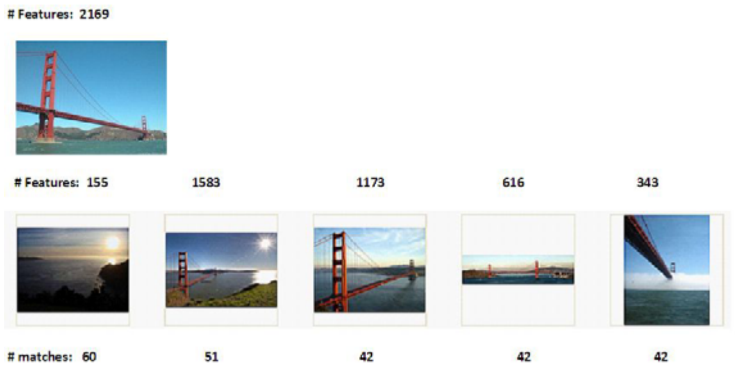
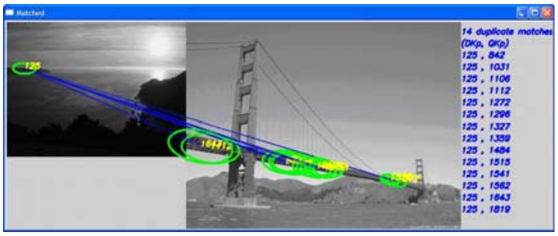


Fig. 4. Top 5 matches retrieved by MAR for the golden gate bridge query



```
14 duplicate matches
(DKp, QKp)
125 , 842
125 , 1031
125 , 1106
125 , 1112
125 , 1272
125 , 1296
125 , 1327
125 , 1359
125 , 1484
125 , 1515
125 , 1541
125 , 1562
125 , 1643
125 , 1819
```

Fig. 5. Right: query image. Left: database image. Illustration of a typical problem of the matching algorithm: one keypoint in the database image has duplicate matches in the query image (14 matches in this example).

to duplicates. Even when these matches are false, they are highly amplified by their multiplicity, which eventually affects the overall retrieval performance. We believe that this problem particularly arises in cases of strong imbalance between the number of keypoints in the database and query images (155 versus 2169). The imbalance forces many keypoints in the query image to match one single point in the database image. The standard ZuBuD data set does not have this issue since its images are more or less uniform and have comparable numbers of keypoints. Contrarily, Wikipedia and web images in general are highly variant, which emphasizes the necessity of having representative sample databases as in Section 3. To solve this problem, our adjusted algorithm prohibits duplicate matches. Whenever a keypoint in the database image has multiple matches in the query image, we only pick the keypoint with the closest descriptor. Applying the new algorithm to the golden gate query, the number of matches between the query image and the previously top 1 image decreases from 60 to 21, and the new top 5 images are shown in Figure 6. We recognize that duplicate matches may still be correct in particular for repetitive structures, however as it is shown for this particular example and in the more extensive results in Section 7, substituting

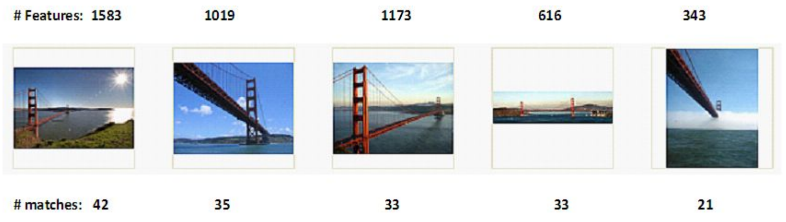


Fig. 6. Top 5 matches retrieved by the adjusted algorithm for golden gate bridge query

the duplicates with the "best" matching descriptor, i.e., the nearest, improves the overall performance of our system.

5 Matching Enhancement by Histograms of Distances

For each keypoint in the query image, the current matching algorithm computes the minimum distance and the second minimum distance between its descriptor and the descriptors of the database image keypoints. Next, the ratio between the distances is used to decide about the match as explained in Section 4.1. Instead of merely relying on the distances ratio, we propose further exploring other statistics based on descriptors' distances. More explicitly, we compute the histogram of minimum distances between the query image and the top ten retrieved database images. Our purpose is to extract more information from these distances about the similarity/dissimilarity between the query and the database images. Next, we examine these histograms in order to remove obviously mismatching images. The cost of this approach is not high, since the distances are already computed and hence we are only leveraging their availability. We applied this approach to 10 query images and their corresponding GPS-constrained databases. We obtained promising results.

5.1 Algorithm

Let Q be the the query image. We first apply our existing matching algorithm and we retrieve the top 10 database images D_1, D_2, ..., D_{10}. Next, we consider each pair (Q, D_i) at a time and build the histogram H_i of minimum distances from keypoints in the query image Q to the database image D_i. There is no additional overhead since these distances were already calculated by the existing matching algorithm. For each histogram H_i, we obtain its empirical mean M_i and and skewness S_i:

$$M_i = \frac{1}{n}\sum_{j=1}^{n} H_{i,j},$$

$$S_i = \frac{\frac{1}{n}\sum_{j=1}^{n}\left(H_{i,j} - M_i\right)^3}{\left(\frac{1}{n}\sum_{j=1}^{n}\left(H_{i,j} - M_i\right)^2\right)^{3/2}}.$$

The smaller the skewness is, the closer to symmetric is H_i. Our main assumptions are:

1. If M_i is large then many of the descriptors pairs between Q and D_i are quite distant and hence are highly likely to be mismatches. Therefore, image D_i must not be considered a match for image Q.
2. If S_i is small (close to zero), then the histogram H_i is almost symmetric. Having many descriptors in Q and D_i that are "randomly" related, i.e., not necessarily matching, would result in this symmetry. We expect this scenario when the the two images Q and D_i don't match and hence there is no reason for the histogram to be biased.

Based on these assumptions, we remove database images that have very low skew S_i. We also cluster the images based on the means M_1, M_2, …, M_{10} into two clusters (we used $k-$means). We remove the images that belong to the cluster with the higher mean. For the experimental results, please refer to Section 7.2. Figure 7 displays the block diagram of our adjusted matching algorithm.

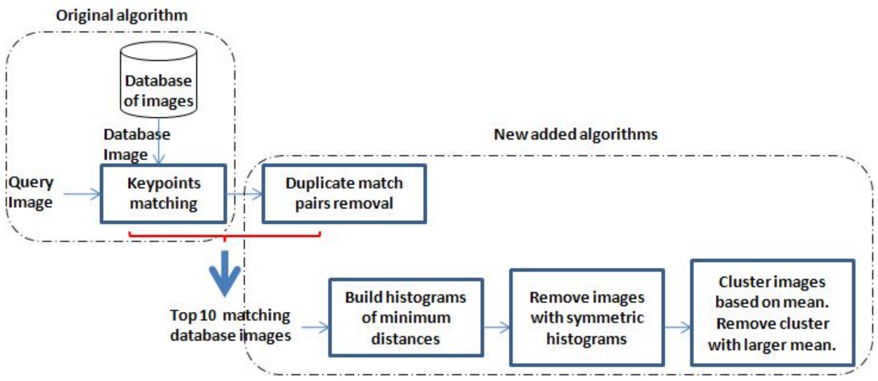

Original algorithm
Database of images
Database Image
Query Image
Keypoints matching
New added algorithms
Duplicate match pairs removal
Top 10 matching database images
Build histograms of minimum distances
Remove images with symmetric histograms
Cluster images based on mean. Remove cluster with larger mean.

Fig. 7. Block diagram of our algorithm for image matching with the duplicates removal and enhancement based on histograms added

6 Matching Enhancement by Database Extension

As mentioned before, the images on our server are downloaded from Wikipedia. Although Wikipedia is a good source for information in general, it is not the best source for images. As we see in Figure 3 there is only one image that matches *the justice of palace* query out of 200 images in the database. However, not only the matching algorithm but also the number of matching images in the database impacts the performance accuracy. Moreover, the query image might be taken from different views and under different light conditions. Having a small number of matching images, e.g., one image in the day light for *the justice of palace* query, would limit the performance of MAR. For this reason, we prefer if the database has more matching images to the query and we aim at extending it by combining GPS-location, geotagged Wikipedia pages, text in these pages, and the large resources of web images. For each Wikipedia page, we don't only download its images, but we also use its title as a text query on Google image search engine. Next, we pick the first few retrieved images and associate them with the same Wikipedia page. As we will see in Section 7.3, the performance of our MAR system improves significantly.

7 Experimental Results

7.1 Duplicates Removal

To test the impact of duplicate matches removal, we ran both the original and the new matching algorithms (see Section 4) on the sample databases of Section 3 for various SURF thresholds. In Figure 8, we display the average of the top 1, top 5, and average precision over the 10 databases. Clearly, removing the duplicate matches improves the performance at almost the same running time.

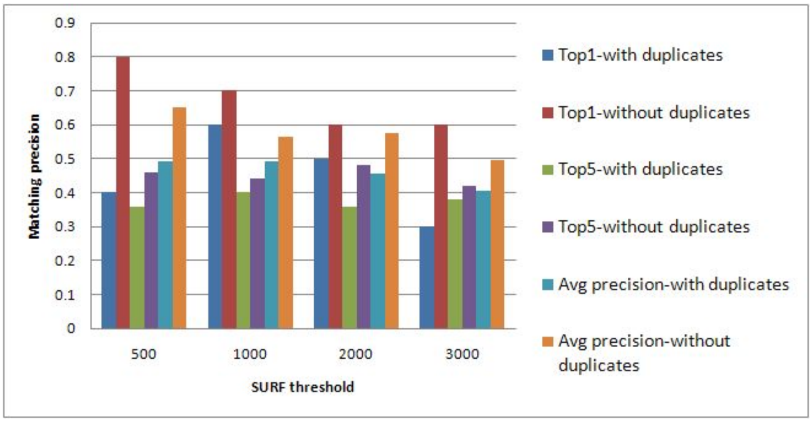


Fig. 8. Top 1, top 5, and average precision results for the 10 landmarks of Section 3 vs. the SURF threshold

7.2 Matching Enhancement through Histograms of Distances

In Figure 9, we display the top 10 retrieved images for each of the 10 databases in Section 3 by the updated matching algorithm of Section 4.2 with duplicates removal. We also used the histograms of distances to further refine the match as explained in Section 5.1. Based on the histograms analysis, images labeled with a red square in the figure are rejected and those tagged with a blue square are not rejected. From these experiments, we see that portraits and statues of people can be clearly distinguished and rejected since most of the queries are those of buildings. Such images in general have a very low skew, i.e., a very symmetric histogram. They also may have a large mean.

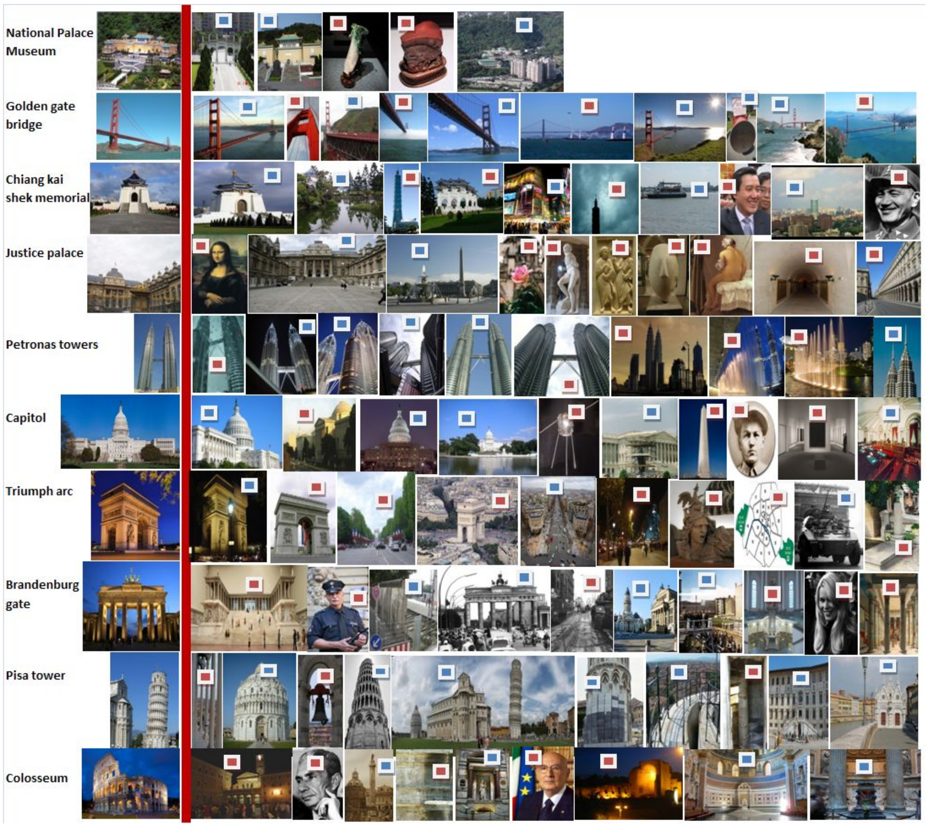


Fig. 9. Results for matching enhancement based on histograms of distances. Images before the red line are the queries.

7.3 Matching Enhancement through Database Extension

For each of our 10 landmarks, we download the top 21 images returned by Google image search when queried with the title of the landmark Wikipedia page and add them to the corresponding landmark database. Next, we apply our improved matching algorithm with no duplicates and using histograms of distances and display the improved results in part (a) of Figure 10. Part (b) illustrates the top 10 images returned by MAR for query images under new light conditions (golden gate bridge at night and triumph arc during the day). The top returned images have similar lighting conditions due to the sensitivity of SURF features to light conditions, which applies to other popular features such as SIFT. Hence we need a diverse set of images in the database in order to compensate for the limitation of visual features. Finally, in the actual scenario Google images relevant to all geotagged Wikipedia pages must be added to our databases, while in part (a) we only added the images corresponding to the query

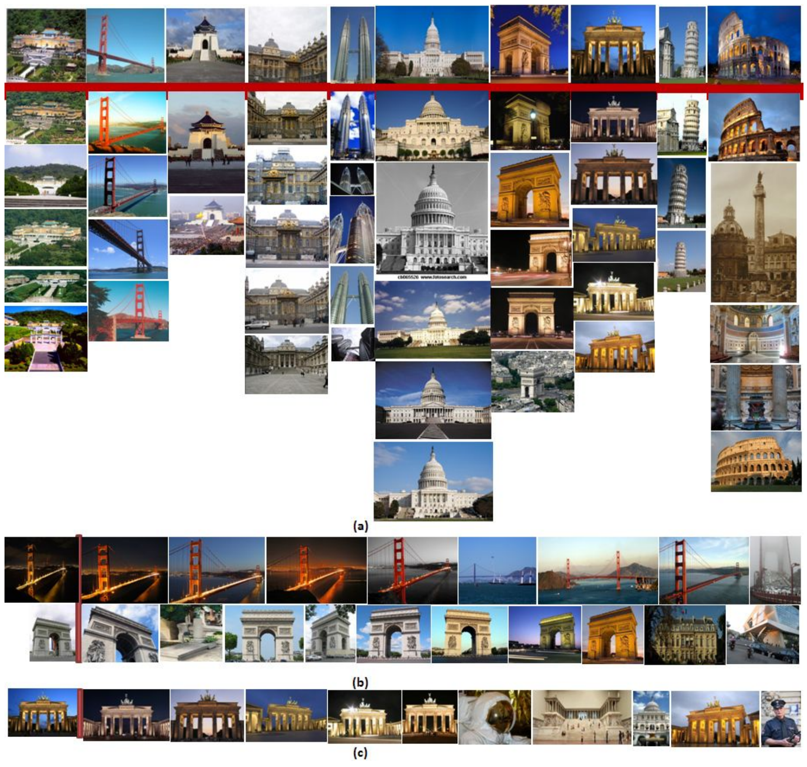


Fig. 10. Results for matching enhancement based on database extension, images before the red line are the queries. (a) All the 10 landmarks. (b) New light conditions for query: night for golden gate bridge, day for triumph arc. (c) Add also a large number of mismatching images to the database (427 mismatching images and 21 google images to 174 database.

landmark. For this reason, in order to make sure that the improvement in the matching accuracy is not due to the bias in the number of matching vs. mismatching images, we added 427 various mismatching images to *the Brandenburg gate* database in addition to the 21 google images (original size 174 images). In part (c), we view the top 10 retrieved images. Clearly, we still have a significant improvement in performance due to the database extension. Note that for parts (b) and (c) we only use the improved matching algorithm excluding duplicates but without refinement based on histograms of distances. Indeed, the main point of the results in part (b) is to stress the sensitivity to light conditions and the need for diverse databases, while the purpose of the results of part (c) is to prove the success of database extension. Both points are clear from the results independently of the particular refinements of the matching algorithm.

8 Implementation and Optimization

8.1 Code Optimization

The original SURF feature extraction code is based on OPENCV implementation (`http://sourceforge.net//projects/opencv/library`) and match algorithm as described in Section 2.2. We further identified the hot spots in the MAR source codes and employed the following optimization:

– Multi-threaded all the hotspots, including interesting point detection, keypoint description generation and image match.
– Data and computation type conversion. In the original implementation, double and float data types are used widely, and floating point computations as well. We quantized the keypoint descriptor from 32-bit floating point format to 8-bit char format. We also converted many floating point computation to fixed point computations in key algorithms. By doing that, not only the data storage is reduced by $4X$, but also the performance is improved by taking advantage of the integer operations. The image recognition accuracy isn't affected from our benchmark results.
– Vectorization. We vectorized the image match codes using SSE intrinsic to take advantage of 4-way SIMD units. Significant speedups are observed compared to original implementation.

8.2 Tracking

The tracking algorithm explained in Section 2.1 has been optimized by: 1) using a simplified multi-resolution pyramid construction with simple 3-tap filters; 2)

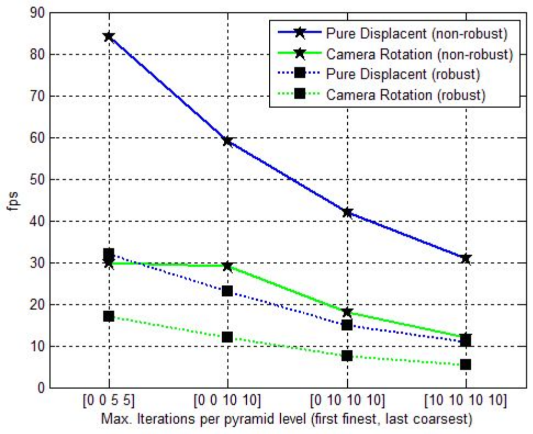


Fig. 11. Performance measured in fps of the image-based stabilization method on an Intel® Atom$^{\text{TM}}$ processor based platform (1.6GHz, 512KB L2 cache) for several choices of model, estimation method, resolution levels and iterations per level

using a reduced linear system with gradients from only 200 pixels in the image instead of from all the pixels in the images; 3) using SSE instructions for the pyramid construction and the linear system solving; 4) using only the coarsest levels of the pyramid to estimate the alignment. Performance was measured on an Intel® Atom$^{\text{TM}}$ system (1.6GHz, 512KB L2 cache) using VGA (640×480) input video and different options. The results are shown in Figure 11, which displays the measured frames per second (fps) for different models (displacement/camera rotation), estimation method (robust/non-robust) and resolution levels and iterations per level used. For pure displacement model, using non-robust estimation and running 5 iterations in the levels 3 and 4 of the multi-resolution pyramid (being level 1 the original resolution) the performance is over 80 fps.

9 Conclusion

In [6], we presented MAR with a fully functional prototype on a MID device with Intel® Atom$^{\text{TM}}$ processor inside. In this paper, we described new improvements to the matching and tracking algorithms in addition to the design of the system and its database. We also built a new realistic testing database. With all these improvements, MAR demonstrates the powerful capabilities of future mobile devices by integrating location sensors, network connectivity, and computational power.

References

1. Pradhan, S., Brignone, C., Cui, J.H., McReynolds, A., Smith, M.T.: Websigns: hyperlinking physical locations to the web. Computer 34, 42–48 (2009)
2. Lim, J., Chevallet, J., Merah, S.N.: SnapToTell: Ubiquitous Information Access from Cameras. In: Mobile and Ubiquitous Information Access (MUIA 2004) Workshop (2004)
3. Zhou, Y., Fan, X., Xie, X., Gong, Y., Ma, W.Y.: Inquiring of the Sights from the Web via Camera Mobiles. In: 2006 IEEE International Conference on Multimedia and Expo., pp. 661–664 (2006)
4. Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W.C., Bismpigiannis, T., Grzeszczuk, R., Pulli, K., Girod, B.: Outdoors augmented reality on mobile phone using loxel-based visual feature organization (2008)
5. Quack, T., Leibe, B., Van Gool, L.: World-scale mining of objects and events from community photo collections. In: Proceedings of the 2008 international conference on Content-based image and video retrieval, pp. 47–56. ACM, New York (2008)
6. Gray, D., Kozintsev, I., Wu, Y., Haussecker, H.: WikiReality: augmenting reality with community driven websites. In: International Conference on Multimedia Expo., ICME (2009)
7. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
8. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)

9. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision, pp. 674–679
10. Nestares, O., Heeger, D.J.: Robust multiresolution alignment of MRI brain volumes, pp. 705–715
11. Nister, D., Stewenius, H.: Scalable Recognition with a Vocabulary Tree. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2006)
12. Shao, H., Svoboda, T., Van Gool, L.: ZuBuD: Zurich Buildings Database for Image Based Recognition. Technique report No. 260, Swiss Federal Institute of Technology (2003)