# Recognizing Locations with Google Glass: A Case Study

Hani Altwaijry and Mohammad Moghimi
Department of Computer Science and Engineering
University of California, San Diego
vision.ucsd.edu

Serge Belongie
Cornell NYC Tech
Cornell University
tech.cornell.edu

## Abstract

*Wearable computers are rapidly gaining popularity as more people incorporate them into their everyday lives. The introduction of these devices allows for wider deployment of Computer Vision based applications. In this paper, we describe a system developed to deliver users of wearable computers a tour guide experience. In building our system, we compare and contrast different techniques towards achieving our goals. Those techniques include using various descriptor types, such as HOG, SIFT and SURF, under different encoding models, such as holistic approaches, Bag-of-Words, and Fisher Vectors. We evaluate those approaches using classification methods including Nearest Neighbor and Support Vector Machines. We also show how to incorporate information external to images, specifically GPS, to improve the user experience.*

## 1. Introduction

The widespread use of mobile devices and smart phones has dramatically increased the applicability and demand for Computer Vision based applications, such as Augmented Reality. The recently introduced *Google Glass*, shown in Figure 2, is an example of a wearable computer that has generated interest in such applications.

Tourists visiting popular locations often entrust a tour guide tell them about the sites they are visiting. The emergence of smart devices initially enabled software tour guides that either deliver information to users based on their current locations [12] or even employed some image analysis to provide information on the contents of the images [41]. However, those applications are limited by the human-computer interaction model. For example, in the case of software tour guides that perform image analysis, the user has to hold the device and point it towards the object of inquiry and take a picture which would then be processed for relevant information. On the other hand, wearable computers change the interaction model dramatically, where the user can query the device for information, as in the case for Google Glass.

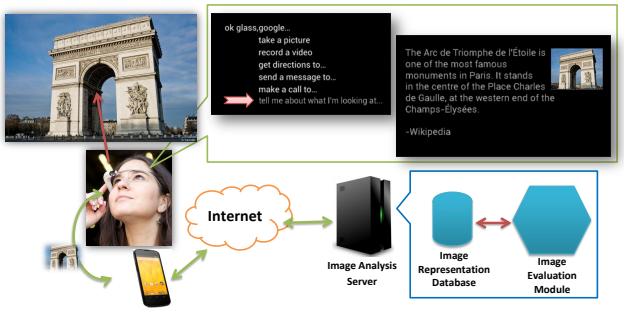


Figure 1. System overview. The user asks the device to inform her about her current view of Arc de Triomphe, and the system responds with the most relevant description in its database.

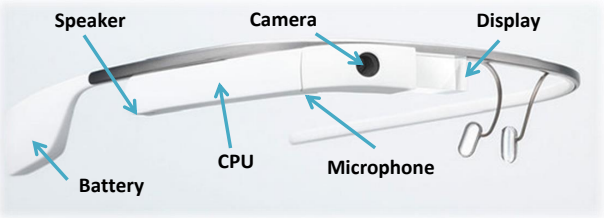


Figure 2. A visual overview of the components in Google Glass.

In this paper, we design a system that delivers a tour guide experience to users of Google Glass. The system's goal is defined as follows: provided an image uploaded by the user's device, and possibly a GPS location, return information about the building in the field of view. Figure 1 gives an overview of the whole system model.

There are a couple of important factors in designing a tour guide system for wearable devices, namely, hardware capabilities and communication. Recent developments in hardware allow for significant processing power in these devices, for example, Google Glass incorporates a dual core 1 GHz processor with 682MB of RAM and a 5 MP camera. These capabilities are recent in the arena of mobile devices, and are essential in delivering a satisfying user experience. Moreover, recent developments in mobile connectivity and

Figure 3. A sample of eight images from the dataset.

data transfer rates help provide a suitable environment for such applications. For example, the recent introduction of 4G LTE technology in the mobile devices market offers rates of up to 300 Mbps in downlink and 75 Mbps in uplink with low latencies [2], where on the other hand, earlier 3G technology could only deliver up to 200 Kbps in downlink, which slightly improved in later revisions to reach downlink rates of 14 Mbps in downlink and 6 Mbps in uplink [1]. Those earlier data transfer rates capped the capabilities of prior systems in delivering a fluid user experience.

### 1.1. Dataset

Multiple works have introduced datasets for landmarks [33], or provided detailed datasets of certain locations such as in [10]. However, to enable empirical on-the-street testing of our system, we collected over 1400 images of 53 buildings and other scenes at UC San Diego. We refer to these views as "locations" or "points-of-interest" throughout the paper interchangeably. The images were GPS tagged with the points-of-interest location as well as the actual capture location. The former was used for training, the latter for testing. We modeled our system as a tour guide for visitors of the campus, where the visitor can ask the device for information about the building in her view. This model allowed us to test the system's performance in real-world circumstances while permitting the use of typical machine learning experimentation standards. Sample images of the dataset are shown in Figure 3.

## 2. Related Work

Our work draws upon ideas from previous works in Mobile Devices [12, 18, 41, 42, 16, 29, 24, 34], Landmark Recognition [39, 20, 28, 27, 40] and Image Geo-localization [37, 38, 19, 13, 21, 4, 3]. Most of these works are essentially performing image retrieval. Our work does not depart from that norm but absorbs it in delivering a functioning system.

### 2.1. Mobile Devices

Since smart mobile devices appeared widely at the end of 1990s, many works have emerged with the objective of performing image retrieval on these devices.

The GUIDE system [12] in 2000 aimed at replicating the tour guide experience via a PC-based tablet device with a dedicated wireless network infrastructure. Their system also leveraged location information via a Differential Global Positioning System (DGPS). Their system aided the user's navigation of a designated area by providing textual and pictorial information on the tablet screen along with maps to help guide the user.

Gausemeier and Bruederlin [18] describe a system from 2003 incorporating an augmented reality device for object recognition. Their system performed recognition by relying on a database of 3D objects which they used to compare against edge-detection results. Although their system is considered primitive by today's standards, it was an effective proof-of-concept and showed the potential of evolving technologies.

Another interesting work in this domain was IDeixis [41]. Developed in 2004, the goal of IDeixis was to provide users of camera-equipped mobile devices location-based information using images. In their system, they follow a server-client model, where the client communicates with the server using Multimedia Messaging Service (MMS). They used Color Histograms and Fourier Transforms in answering queries. Their average turnaround time was 25 seconds per query.

Another more recent work, from 2008, is [24], in which the system compares images uploaded from a smart phone along with the current GPS location to a dataset of panoramic images using SIFT features. The system then determines the pose of the user and returns relevant information.

### 2.2. Landmark Recognition

In contrast to the previous section, research involving landmarks [27, 28, 40] has generally focused on the recognition task itself, without regard for an end-user application. A survey on Landmark Recognition for mobile devices is given in [11].

In another work from 2008, Li *et al*. [27], harvested images from Flickr and adopted the Bag-of-Words model in representing those images. They also leveraged textual and temporal data associated with the landmark images which they treated as image-tags. To perform recognition, a multiclass Support Vector Machine [14] was trained treating each landmark as a separate class.

In [40], Zheng *et al*. downloaded images from Picasa and Panoramio, and extracted features from them using Gabor filter-bank responses. Their recognition step uses Nearest Neighbor to pick matches. Moreover, since they exhaustively match against all images, their complexity is heavily reduced through the use of k-d trees [6]. A match is found when the nearest neighbor satisfies a given threshold.

### 2.3. Image Geo-localization

In a similar spirit to Landmark Recognition, Image Geo-Localization attempts to determine the location of a given image by comparing it against a large database of images with known locations. This is inherently not different than landmark recognition, except that in this scenario, we are comparing to a much larger database with arguably many more locations.

Schindler *et al*. in [37] worked with data encompassing about 20 kilometers of driving distance amounting to 30 thousand images. Zamir *et al*. [43] went further and worked with 100,000 images from Google Street View. Having a large number of images led both of them to seek approximate approaches for feature matching, such as Vocabulary Trees in [37] and a k-d tree in [43]. Both used variants of nearest-neighbor to count matching features as the basis of their recognition step. Hays *et al*. in [19] went even further, employing over 6 million images obtained from Flickr. Their goal was to localize images over the entire planet. Features included Color Histograms, Texton Histograms [25] and GIST [32]. They also used a variant of nearest neighbor, namely $k$-NN, in combination with mean-shift clustering to localize the images.

In [10], Chen *et al*. collected 1.7 million training panorama images with detailed tags, such as the GPS and viewing direction, using a vehicle from the city of San Fransisco, and 803 test images from mobile devices. Their panoramas were converted to perspective images by leveraging information in the image tags. Those perspective images are split into two databases, that store a vocabulary tree for query processing. In handling their queries, they use SIFT with Term Frequency-Inverse Document Frequency (Tf-Idf) weighting and compare using and not using the GPS coordinates in filtering their results. In contrast to their work, we aim to provide information about certain locations, i.e., in a tour-guide manner, and not localize the user's coordinates, and hence we follow a classification approach.

The Bag-of-Words model was used by Cao *et al*. in [8] with SIFT features. They trained linear SVMs corresponding to clusters of images that belonged to certain locations, which were obtained via a clever vertex cover approach on a graph built by relating images together using feature matching in an earlier step.

## 3. Approaches to Recognition

We have used multiple approaches to data modeling and evaluation in our system to facilitate our study of different aspects including location recognition accuracy, training time and response time. We have implemented the following methods on our server-side: Holistic Approaches, (e.g. GIST [32]), Bag-of-Words with different feature descriptors, and Fisher Vectors with different feature descriptors, evaluating using Nearest Neighbor approaches and Support Vector Machines. All of those methods were designed to work on the server-side answering queries presented by Google Glass.

### 3.1. Holistic Approaches

Holistic approaches calculate a whole-image descriptor, and they have been widely used in scene recognition [32]. We use GIST [32], Color Histograms, and Histograms of Oriented Gradients (HOG) [15] as holistic representations of the images in our database. Although HOG is widely used in the literature of object detection, it can also be used in a holistic manner, and therefore we incorporated it in our system. The end result of these methods is a single vector that describes the image that gave rise to it.

### 3.2. Bag-of-Words

The Bag-of-Words (BoW) model is a widely used approach in Computer Vision for various tasks including object recognition and retrieval. In this work, we use the BoW model based on several descriptor types, namely, SIFT [30], SURF [5] and BRISK [26].

To find visual words, we use K-Means [31] for a hard-assignment codebook as in the typical BoW model. The final image representation is a vector of dimensionality equal to the codebook size. Furthermore, we used Tf-Idf weighting [35] in some instances for comparison against basic BoW as will be discussed in Section 4.

### 3.3. Fisher Vectors

Fisher Vectors [36] have recently achieved state-of-the-art results on many object detection datasets, such as Pascal VOC. They have not previously been considered in the case of landmark and location recognition. We used dense PHOW [7] features and SIFT [30] in two flavors, one using Spatial-Pyramid-Matching [23], and one without. We refer the reader to [36] for the details of using and implementing Fisher Vectors.

### 3.4. Evaluation Methods

All of the previously mentioned data representation approaches produce a vector describing the whole image. Those vectors constitute the training data for our evaluators. Those evaluators follow standard approaches, namely: $k$-Nearest Neighbor, One-vs-All Support Vector Machine, and a One-vs-One Support Vector Machine. We will discuss this further in 4. After the generation of model data, depending on the chosen representation, we store all model information in a Spatial SQL Database for the evaluation and query step.

We test the evaluation methods under two modes, *GPS-active* and *no-GPS*. In the GPS-active mode, we extract the

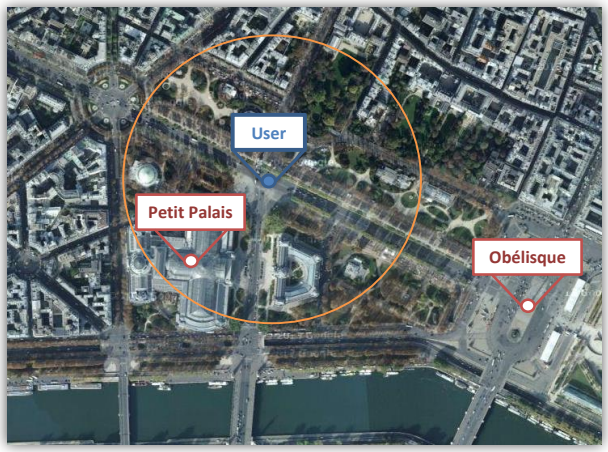


Figure 4. An illustration of how we use the GPS location in the GPS-active mode to reduce the search space of possible points of interest. Here, "Petit-Palais" is returned as a viable point-of-interest to compare against the query. The figure illustrates an imaginary setting in Paris, France.

GPS location data from the EXIF-tag of the JPEG file, or supply it separately with the client request. The GPS location is used in a Spatial-SQL query to the spatial database to retrieve the possible locations around the user. This step reduces the number of candidate locations, and therefore, the number of evaluations we have to make. On the other hand, in the no-GPS mode, we query the database and retrieve all stored models for evaluation. Figure 4 gives an illustration of how the Spatial Database is used to reduce the search space of the points-of-interest.

### 3.5. Deep-Learning

Deep Belief Networks have made an impact recently on a number of commonly used benchmarks. The Convolutional Deep Belief Network introduced by Krizhevsky *et al.* in [22] achieves the state-of-the-art results on the challenging ImageNet [17] object classification dataset.

We have attempted using Convolutional Deep Belief Network, however, our networks performed poorly on our test instances. This was caused by the small number of training images compared to the huge number of parameters that need to be tuned for a network with a similar structure to the one used in [22]. This effectively led to networks that remembered each training image, and henceforth we decided to leave these approaches for future work.

## 4. Experiments

Testing our system was accomplished in two settings. In the first setting, we divided the dataset into 1204 training images and 272 testing images. We note that the test images were captured at a different time and day. We evaluated each method on the split separately, and measured the accuracy, timing for feature extraction and evaluation on the same computer. On the other hand, in the second setting, we empirically test the system by evaluating on 66 images from Glass. We tested this setting by choosing two good representative methods as observed from results on experiments in the first setting.

### 4.1. Google Glass Details

Our system is mainly designed to work with Google Glass, although it is possible to support other smart phones. We developed a Google Glass Android application that is mainly concerned with the operation of answering the user query of "tell me about what I'm seeing."

When the application receives the voice command from the user, it captures an image from the camera and sends the data through a WiFi hotspot created by a companion mobile device to the server over a cell network operating under LTE data speeds. Once the response is received, the captured image is displayed overlayed with textual information about the location that is also spoken at the same time.

### 4.2. Server Details

Our demo server had an Intel i7 Processor with 12GB of RAM, running the Ubuntu operating system. The web service is implemented in Java under the "Play" WebApp Framework integrated with OpenCV, LIBSVM [9], and MySQL. Queries are received via HTTP requests with attached images, and an optional GPS location. The image is extracted from the request and is examined for EXIF GPS meta-data or the provided GPS data is used if it exists, otherwise the image is searched in the entire database. Image processing then follows according to each described approach. The final evaluation step uses the GPS information whenever possible.

### 4.3. Experimentation Details

The experimental setup was different than the demo server. The different approaches (except for Fisher Vectors) were evaluated on multiple machines running on Amazon EC2 and using the Amazon S3 storage services. EC2 instances differed in the amount of RAM provided, which ranged from 16GB to 32GB as required by the different methods to accommodate the large numbers of features extracted.

In our holistic approaches, we extracted GIST, HOG and Color Histograms. For GIST, we resize all images to $128 \times 128$ regardless of the aspect ratio as in [32] and extract a 960-dimensional vector for each image. In the case of HOG, we resize all images to $128 \times 96$ maintaining the aspect ratio of 4:3 and cropping images if needed to achieve the mentioned aspect ratio. This is done in contrast to GIST to preserve the structure in the image. We have $8 \times 8$ pixel cells inside $2 \times 2$ blocks, which overlap on cell boundaries.

The result is a vector of $5940$ dimensions which we normalize with $L_2$ norm. As for Color Histograms, we extract a 36-dimensional histogram of RGB colors, which is $L_2$ normalized. All vectors are then directly used in kNN and SVM evaluations.

For our experiments with the BoW model, we used three settings of the codebook size, $K = \{100, 1000, 2000, 3000\}$, for all feature types SIFT, SURF, and BRISK. We used the $L_2$ norm in all instances. The codebooks generated were used as well in the Tf-Idf weighted BoW scenario. We computed the Term Frequency as:

$$TermFrequency_i = \frac{RawFrequency_i}{\max\left(RawFrequency_j\right) \forall j \in d} \tag{1}$$

where $i$ is a visual word, $RawFrequency_i$ denotes the raw count of the word $i$ in the image $d$. Furthermore, the Inverse Document Frequency (IDF) was calculated as:

$$IDF_i = \log \frac{|D|}{\sum_i RawFrequency_i} \tag{2}$$

where $|D|$ denotes the number of images in the dataset. In either case of BoW, the vector for each image is stored in the database for retrieval during query time. Lastly, we $L_2$ normalize the vectors.

A hybrid approach was also pursued by combining both holistic and BoW approaches through fusing BoW, GIST and Color Histograms into one vector. Each component vector was $L_2$ normalized separately before fusing those together. We called those "Combined-1" made by fusing 2000-TFIDF SURF+GIST+Colors Histogram, and "Combined-2" made by fusing 2000-BoW SURF+GIST+Colors Histogram.

For Fisher Vectors, we trained 256 Gaussians for the Gaussian Mixture Model for both the SIFT and PHOW features. For SIFT, we reduced the dimensionality to 64 using PCA, whereas for PHOW, we reduced the dimensionality to 80. We also used sum-pooling in the Spatial-Pyramid. We tested the Fisher Vectors with One-vs-All SVMs only, due to the impracticality of performing exact k-NN or training $\binom{n}{2}$ SVMs for the One-vs-One SVM case, as the Fisher Vectors are fairly large (327680 dimensions for PHOW+SPM).

To leverage the GPS information, we construct a circle of 250 feet in radius, and query the database for locations within a minimum bounding rectangle of that circle. We use the minimum bounding rectangle because it is more efficient in querying the database than using an arbitrary polygon. To guard against having the database return an empty result set, we iteratively query the database with a larger radius until a valid result set is returned; this effectively widens the horizon of the search.

Note that for experimentation purposes, all vectors were loaded by reading the database once, however, with a very large database this may not be possible, and vectors would have to be loaded per-query to save memory. This should not degrade response time in the GPS-active mode because typically a limited number of locations is returned.

## 4.4. Results

### 4.4.1 $k$-Nearest Neighbor Results

For k-Nearest Neighbor, we return the $k$ most similar image representations, and provide the label of those images, i.e. for $k = 5$ we return the result of the first five most similar images, and their respective labels. We consider the result correct if any of those labels match the true label.

The results for $k$-NN are given in Figure 5. In the case of no GPS assistance, the best method for top-1 accuracy was BoW with SURF features using a codebook of 3000 visual words achieving 63%. The case was similar for the top-5 accuracy. The combination of BoW SURF-2000, GIST, and Color Histograms performed worse than the main BoW SURF-2000 representation, however, it showed more promising results for the top-5 accuracy. Moreover, the Tf-Idf weighted SURF appeared to suffer when the codebook size was increased to 3000 as opposed to the generic BoW representation, which could be caused by the number of visual words found in the image being less than the codebook size which in turn gets affected by the weighting process.

We observed that with incorporating GPS information, the performance is greatly improved by a factor close to 50% in the worst case of BRISK. This mainly shows how GPS information can greatly improve the result by comparing against images in a local neighborhood. However, we notice that the performance is saturated for the top-5 accuracy using GPS just under 90%. This indicates how the GPS could mislead the system if the object of the image is located farther away than 250 feet.

In general, BRISK features performed the worst which essentially showed that they are not well suited for this problem. The main advantage of BRISK is fast computational speed as Table 1 shows an overall response time for queries in under 1 second. However, the case is still challenged by GIST which gives good performance for a very low cost of feature calculation, especially in the case using the GPS as Figure 5 shows. The BoW SIFT performed better than the Tf-Idf weighted version, and while both had their performance degrade with increase codebook size, we noticed that the unweighted version degraded at a lower rate. Furthermore, we noticed that HOG performed badly as a holistic descriptor in all cases barely overcoming BRISK.

We computed the average feature extraction times and average response times for each query as shown in Table 1. The quickest response time was achieved by HOG, followed by GIST then BRISK. Out of those three, GIST seemed the most reliable, while still performing worse than SIFT

and SURF. Moreover, SURF was about 1.25 seconds slower than SIFT in the feature extraction time explaining the delayed response. This draws the case for opting to use SIFT instead of SURF if a quicker response is needed, especially since the GPS-active case has a high accuracy for both features.

#### 4.4.2 SVM Results

For the SVM evaluation methods, we implemented both One-vs-All and One-vs-One multi-classification approaches using Linear SVMs that were trained with 5-fold cross-validation on the training set. Figures 6 and 7 show the results for both cases.

The top-1 accuracy of 87%, and top-5 accuracy of 92% achieved by Fisher Vectors with PHOW features and Spatial Pyramid Matching. We observed that Fisher Vectors with PHOW dense features performed a lot better than using sparse SIFT features, which reflects how dense features help in capturing the overall scene.

With respect to the other approaches, the One-vs-All approach worked better in this case as seen by the top-1 accuracy of 77%, and top-5 accuracy of 87% achieved by Combined-2 as opposed to 69% and 85% in the One-vs-One SVM implementation. The case was also similar for the GPS-active mode.

SVMs generally performed better than $k$-NN, except for certain cases such as with SIFT BoW. The response times for SVMs, shown in Table 1 were comparable to the case of $k$-NN. The response times were averaged across both GPS settings. We believe this similarity in response times is due to the fact that the SVM stores some of the support vectors, which are image representations, and since locations do not have a lot of examples, most of those will be stored by the SVM, and hence the timing is very comparable to $k$-NN which evaluates on all images for a given location[1]. The timing for Fisher Vectors with PHOW was prohibitively slow compared to using SIFT.

We observed that in the GPS-active case, the top-5 accuracy was saturated for both SVM cases just under 90%, which the Fisher Vectors outperformed using PHOW+SPM. This clearly indicated that a limited GPS search could be misleading, however, no fatally. In separate experiment not shown in the figures, we used Fisher Vectors with PHOW+SPM, and extended the GPS-search horizon to include at least 5 targets. In this case, the GPS-active method achieved 92% accuracy the top-1 candidate, and 98% for top-5 accuracy.

[1]LIBSVM stores the support vectors for the Linear SVM, and does not store the hyperplane as it solves the convex optimization in the dual space.

| | **Top-1 Accuracy** | **Top-5 Accuracy** |
|---|---|---|
| Combined-2 | 51.5% | 74.2% |
| FV SIFT | 71.21% | 80.3% |

Table 2. Accuracy of answering Glass queries *without GPS*.

### 4.5. Results on Google Glass

We tested our system on 66 images taken with Google Glass using Fisher Vectors based on SIFT and the Combined-2 feature blend evaluating on a One-vs-All SVM *without GPS* assistance. The accuracies achieved are given in Table 2. We measured the average processing time for the whole query-response on a prototype Android device running on the Verizon LTE network, which corresponded to roughly 2-4 seconds. The features for the response time experiment were Tf-Idf SIFT features. Unfortunately, we were unable to test Glass images with GPS information in the physical experiment due to difficulties faced by the Glass device in capturing GPS data from the companion mobile device.

## 5. Conclusion

In conclusion, we have designed and implemented a functional system that accepts queries from Google Glass and delivers information on what a user is seeing. Our system achieves 87% top-1 accuracy without GPS and 92% with GPS on the test-set using Fisher Vectors on PHOW features with One-vs-All SVMs. Running the system with Fisher Vectors on SIFT features performs significantly faster and the accuracy on Glass without using the GPS is promising at 71%. Our tests suggest that the performance on Glass will improve when running in the GPS-active mode. Our future goals include scaling the system to handle more images efficiently, where we hope ultimately to leverage the spatial database for improved responsiveness and accuracy. We are also planning on optimizing the database query horizon to overcome the saturation faced by the methods in this work.

## Acknowledgments

We would like to thank all of our colleagues at the Computer Vision group at UCSD CSE for their valuable discussions and comments. This work was supported by the Google Focused Research Award and by the KACST Graduate Studies Scholarship.

## References

[1] Universal Mobile Telecommunications System (UMTS), UE Radio Access capabilities(3GPP TS 25.306 version 6.13.0 Release 6) . Technical report, European Telecommunications Standards Institute, 2009.

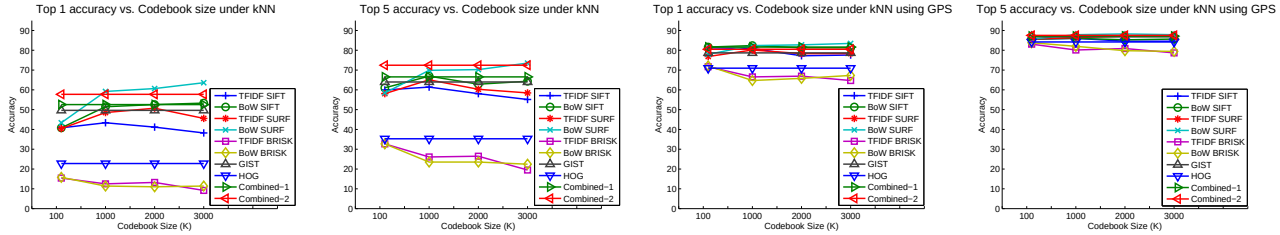


Figure 5. The accuracy of the different approaches as the Codebook size varies. The evaluation method was kNN. Note that GIST, HOG, Combined-1, and Combined-2 do not have codebooks, and their result is constant. Combined-1: 2000 TFIDF SURF+GIST+Colors Histogram, Combined-2: 2000 BoW SURF+GIST+Colors Histogram.

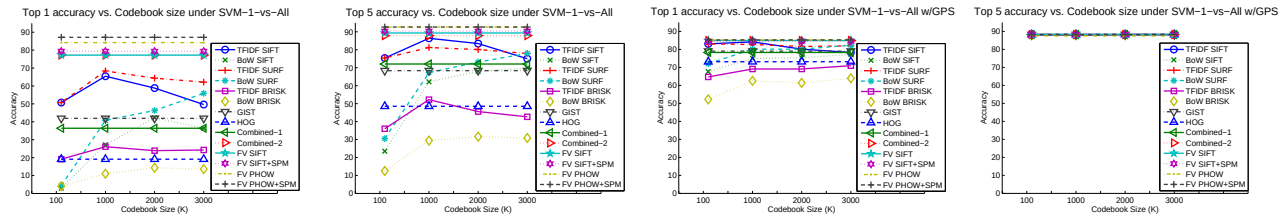


Figure 6. Accuracies for evaluation under One-vs-All SVMs across the different feature types. Note that GIST, HOG, Combined-1, and Combined-2 do not have codebooks, and their result is constant. Refer to Figure 5 for details.


[2] D. Astely, E. Dahlman, A. Furuskar, Y. Jading, M. Lindstrom, and S. Parkvall. Lte: the evolution of mobile broadband. *Communications Magazine, IEEE*, 2009.

[3] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *ECCV*, 2012.

[4] M. Bansal, H. S. Sawhney, H. Cheng, and K. Daniilidis. Geo-localization of street views with aerial image databases. In *Multimedia*, 2011.

[5] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[6] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 1975.

[7] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *ICCV*, pages 1–8, Oct 2007.

[8] S. Cao and N. Snavely. Graph-based discriminative learning for location recognition. In *CVPR*, 2013.

[9] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011.

[10] D. M. Chen, G. Baatz, K. Koser, S. S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, et al. City-scale landmark identification on mobile devices. In *CVPR*, 2011.

[11] T. Chen, K. Wu, K.-H. Yap, Z. Li, and F. Tsai. A survey on mobile landmark recognition for information retrieval. In *Mobile Data Management*, 2009.

[12] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a context-aware electronic tourist guide: Some issues and experiences. In *SIGCHI*, 2000.

[13] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, 2008.

[14] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[16] N. Davies, K. Cheverst, A. Dix, and A. Hesse. Understanding the role of image recognition in mobile tour guides. In *MobileHCI*, 2005.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[18] J. Gausemeier, J. Fruend, C. Matysczok, B. Bruederlin, and D. Beier. Development of a real time image based object recognition method for mobile ar-devices. In *AFRIGRAPH*, 2003.

[19] J. Hays and A. A. Efros. im2gps: estimating geographic information from a single image. In *CVPR*, 2008.

[20] J. Hu, J. Sawyer, and J.-Y. Herve. Building detection and recognition for an automated tour guide. In *Systems, Man and Cybernetics*, 2006.

[21] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009.

[22] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[23] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[24] J. Lee, K.-C. Yow, and A. Sluzek. Image-based information guide on mobile devices. In *ISVC*, 2008.

[25] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 2001.

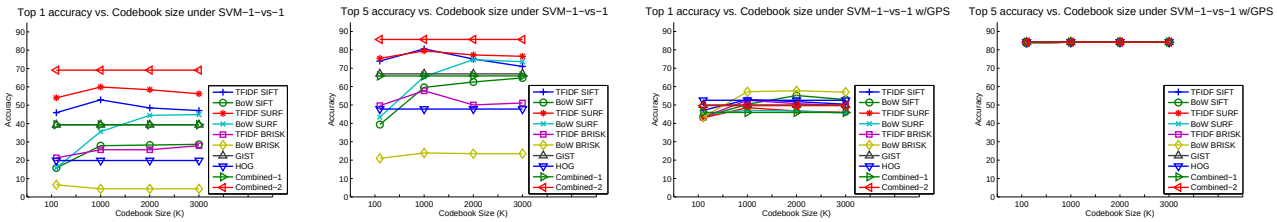


Figure 7. Accuracies for evaluating One-vs-One SVMs across the different feature types. Note that GIST, HOG, Combined-1, and Combined-2 do not have codebooks, and their result is constant. Refer to Figure 5 for details.

| Feature | Avg. Feature Extraction | Avg. kNN | Avg. 1vA | Avg. 1v1 |
|---|---|---|---|---|
| TFIDF SIFT | 1.52 | 2.24 | 2.17 | 2.14 |
| BoW SIFT | 1.42 | 2.34 | 2.15 | 2.12 |
| TFIDF SURF | 2.94 | 3.57 | 3.67 | 3.66 |
| BoW SURF | 2.87 | 3.58 | 3.74 | 3.72 |
| TFIDF BRISK | 0.58 | 0.8 | 0.84 | 0.81 |
| BoW BRISK | 0.48 | 0.81 | 0.83 | 0.80 |
| GIST | 0.52 | 0.51 | 0.55 | 0.56 |
| HOG | 0.4 | 0.44 | 0.72 | 0.61 |
| Combined-1 | 3.52 | 4.34 | 4.49 | 4.44 |
| Combined-2 | 3.45 | 4.35 | 4.54 | 4.48 |
| FV SIFT | 0.87 | - | 0.92 | - |
| FV SIFT+SPM | 0.91 | - | 0.95 | - |
| FV PHOW | 9.36 | - | 9.41 | - |
| FV PHOW+SPM | 17.05 | - | 17.12 | - |

Table 1. Average (with and without GPS) timing for the different features while using kNN, One-vs-All SVMs and One-vs-One SVMs. All times are in seconds. Feature extraction times for "Combined-1" and "Combined-2" is the sum of their components' feature extraction time. Fisher Vector times are reported as on the demo server.


[26] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[27] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV*, 2008.

[28] Y. Li, D. Crandall, and D. Huttenlocher. Landmark classification in large-scale image collections. In *ICCV*, 2009.

[29] J.-H. Lim, Y. Li, Y. You, and J.-P. Chevallet. Scene recognition with camera phones for tourist information access. In *Multimedia and Expo*, 2007.

[30] D. G. Lowe. Object recognition from local scale-invariant features. *ICCV*, 1999.

[31] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[32] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001.

[33] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[34] W. Premchaiswadi, A. Tungkatsathan, and N. Premchaiswadi. Mobile image search for tourist information using accc algorithm. In *Personal Indoor and Mobile Radio Communications*, 2010.

[35] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*.

[36] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *IJCV*, 2013.

[37] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007.

[38] G. Schindler, P. Krishnamurthy, R. Lublinerman, Y. Liu, and F. Dellaert. Detecting and matching repeated patterns for automatic geo-tagging in urban environments. In *CVPR*, 2008.

[39] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*, 2006.

[40] Y. tao Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, R. Bissacco, O. Brucher, T. seng Chua, and H. Neven. Tour the world: Building a web-scale landmark recognition engine. In *CVPR*, 2009.

[41] K. Tollmar, T. Yeh, and T. Darrell. IDeixis–searching the web with mobile images for location-based information. In *MobileHCI*. 2004.

[42] T. Yeh, K. Tollmar, and T. Darrell. IDeixis: image-based deixis for finding location-based information. In *CHI*, 2004.

[43] A. R. Zamir and M. Shah. Accurate image localization based on google maps street view. In *ECCV*, 2010.