A memory efficient discriminative approach for location aided recognition

Varsha Hedau[1], Sudipta N. Sinha[2], C. Lawrence Zitnick[2], and Richard Szeliski[2]

[1]Nokia Research, Sunnyvale, CA, USA
varsha.hedau@gmail.com
[2]Microsoft Research, Redmond, WA, USA
{sudipsin,larryz,szeliski}@microsoft.com

Abstract. We propose a visual recognition approach aimed at fast recognition of urban landmarks on a GPS-enabled mobile device. While most existing methods offload their computation to a server, the latency of an image upload over a slow network can be a significant bottleneck. In this paper, we investigate a new approach to mobile visual recognition that would involve uploading only GPS coordinates to a server, following which a compact location specific classifier would be downloaded to the client and recognition would be computed completely on the client. To achieve this goal, we have developed an approach based on supervised learning that involves training very compact random forest classifiers based on labeled geo-tagged images. Our approach selectively chooses highly discriminative yet repeatable visual features in the database images during offline processing. Classification is efficient at query time as we first rectify the image based on vanishing points and then use random binary patterns to densely match a small set of downloaded features with min-hashing used to speedup the search. We evaluate our method on two public benchmarks and on two streetside datasets where we outperform standard bag-of-words retrieval as well as direct feature matching approaches, both of which are infeasible for client-side query processing.

1 Introduction

The ubiquity of cameras on GPS-enabled mobile devices nowadays makes it possible to visually query the identity of a specific landmark in a scene simply by taking a picture and uploading it to a server for processing. The feasibility and accuracy of such applications is rapidly improving as geo-tagged image databases such as Flickr, Google and Bing Maps grow by the day. Recognizing landmarks reliably in streetside photos however poses some challenges. First, streetside buildings exhibit great variations in appearance due to changes in viewpoint, illumination, weather, seasons or even due to changes in the scene structure. Second, clutter in the scene due to the presence of people, vehicles etc can be unavoidable when issuing a query. Finally, not all streetside buildings are easy to recognize. In fact some can be quite difficult to distinguish from one another due to their similar appearance and presence of ambiguous visual features.

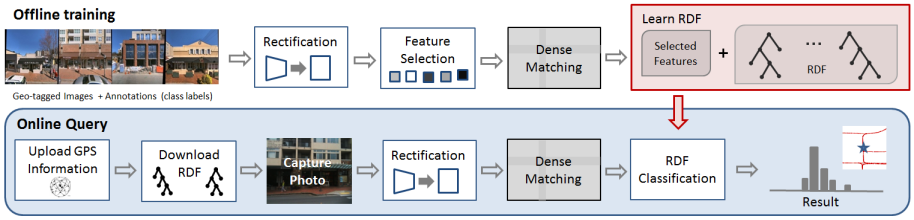

Fig. 1. Overview of our location-aided recognition approach. A device using our method will upload only GPS location to the server at query time; a compact, location specific classifier trained offline will be downloaded to the device and evaluated on the client.

Several recent approaches to location recognition [4, 12, 15, 18, 17, 19, 22, 23, 28, 29] use the query image to retrieve similar images from a geo-registered image database using robust feature matching based on interest points and local feature descriptors. The use of quantized descriptors and bag-of-words (BoW) models are popular due to their scalability [9]. These retrieval methods however require significant memory as they often rely on re-ranking based on a geometric verification step that requires storing all the database image features. When recognition is performed on a server this is less of a concern. However, the latency associated with uploading an image to the server over a slow network can be quite significant. To reduce latency of landmark recognition on mobile devices, we explore an alternative approach where location specific data is downloaded to the device based on its GPS location after which all computation happens on the device. To make this approach feasible the download size must be small and the subsequent processing must be efficient on low-end devices. Our new approach addresses these pertinent issues and we analyze the accuracy and download size tradeoffs of our method and existing methods under these constraints.

In this paper, we propose a memory efficient discriminative approach to landmark recognition that uses the approximate GPS coordinates of the querying device. Instead of storing all the images in a database, we selectively store information necessary to uniquely distinguish each landmark from other landmarks within a reasonable geospatial scope. This is accomplished by training a Random Decision Forest (RDF) classifier [6] to classify each query to the set of possible locations or landmarks within a small region surrounding the GPS location of the querying device. During offline processing, local image features reliably matched across training images of the same landmark are first automatically discovered and matched densely across every training image, yielding features that can be used for landmark classification. During the training stage, a small set of discriminative features are automatically selected from this pool of potential features. For efficient dense matching[1] and for invariance to perspective distortion, all images are rectified prior to matching [4, 22] (see Figure 1 for an overview). For fast query processing, we propose using min-hash to further accelerate the dense matching step. We demonstrate our approach on two public benchmarks for landmark recognition and in two urban scenes where all query images are

[1] a patch is compared to patches at all 2d positions across a range of discrete scales

captured on mobile devices. The number of landmarks in these datasets vary between 50–200. In all four cases, our classifiers are quite compact with download size in the range of 150–230 KBytes, with accuracy comparable or better than existing methods which are also impractical for on-device processing.

Our main contribution is a new approach suitable for landmark recognition on the mobile device. We train compact random forest classifiers using geospatial scope of the training images and in the process a small set of discriminative and repeatable local features are discovered. At query classification time, these features are densely matched in the image using binary descriptors. A min-hash technique significantly improves the efficiency of this dense matching step.

1.1 Related work

Most prior work in landmark recognition has focused on improving keypoint-based retrieval. Knopp et al. [15] remove confusing features in a bag-of-words model, while Li et al. [19] prioritize the matching of repeatable and frequently occurring features. Turcot and Lowe [26] remove features that are not repeatable and merge the remaining features from neighboring images. A vocabulary tree based approach that maximizes the information gain of the visual words was proposed by [23] whereas Jegou et al. [14] developed compact global descriptors for scalable similar image retrieval. Zamir et al. [28] remove noisy matches between images using a variant of the descriptor distance ratio test, and Zhang et al. [29] uses robust motion estimation. Our approach avoids keypoint extraction and instead uses dense matching of a few selective features in the whole image across position and scale. Other methods perform location recognition using scene categorization [12], or using structure from motion point clouds [13, 19].

Image rectification is important for wide baseline matching [22, 29, 5] and also useful for location matching with upright SIFT features [4]. Our work is closely related to [18] that uses a discriminative approach for classifying landmarks using SVMs with histograms of visual words as features. However they rely on text features for obtaining higher classification accuracy. In contrast, we use random decision forest classifiers [6] which are ideal for multi-class classification[2, 16, 25], can be compactly represented and allow fast evaluation at classification time.

Compact feature descriptors have been proposed for feature extraction on a mobile device [8] and a number of approaches upload a set of feature descriptors instead of the query image [3]. Although this improves the efficiency of the upload, significant latency may still be present. Our use of min-hash for accelerating the dense matching is related to prior work on min-hash for efficient retrieval of near duplicate images from large databases [11, 10].

2 Learning location classifiers

We recognise the landmark seen in a query image by classifying it as one of the predefined landmark classes. Our training set consists of geo-referenced images annotated with locations labels, typically corresponding to individual buidlings or street intersections. We use random decision forests (RDF) for this multi-class classification task. The input features for RDF are computed from a set of

selected patch templates that are densely matched across the image. To ensure that discriminative features are selected during training, we first identify patches that are repeatable within each landmark class [26]. We rectify the images during preprocessing to remove perspective distortion which also makes dense matching more efficient. Our approach is now described in detail.

2.1 Rectification

Fig. 2. Three query images and their rectified versions computed by our method.

Planar building facades often undergo severe perspective distortion, based on the camera's orientation and position. Searching over all plausible distortions is computationally prohibitive for dense matching. If the camera's focal length is known and vanishing points can be identified, the degrees of freedom can be reduced using image rectification. Dense patch matching on the rectified image reduces the search to 2d position and scale. Rectification has been used similarly in prior work [4, 22]. Our automatic metric rectification method first detects orthogonal vanishing points (VP) using an approximate focal length estimate. By assuming that images have small roll angle, we can easily identify the vertical VP in the image. As in *upright* SIFT [4], the lack of rotational invariance makes our features more discriminative.

We detect vanishing points in multiple stages. First, the vertical VP is estimated via sequential RANSAC on 2D line segments subtending a small angle to the vertical. For speed and accuracy, longer lines are given preference during the random sampling. When the focal length is known, the vertical VP determines the horizon line in the image. Horizontal VPs are then found using a 1-line RANSAC on non vertical lines that intersect the horizon. When the focal length is unknown, our RANSAC hypothesis also includes a random guess of the focal length, sampled from the normal distribution $N(f, \sigma)$ where $f = 1.5$ is the normalized focal length and $\sigma = 1.0$. If two orthogonal vanishing points are found, we rectify the image using the 2D homography, $H{=}KR^{-1}K^{-1}$ where, the matrix $K = \text{diag}([f f 1])$ represents camera intrinsics with normalized focal length f and R denotes the 3D rotation with respect to the 3D vanishing directions. The image quad to be rectified is chosen based on a threshold for the maximum distortion induced by H. Figure 2 shows some examples of images rectified using our approach. However, an accurate rectification is not essential for our approach. When only the vertical VP is detected, we only perform roll correction, thereby eliminating one degree of freedom in camera rotation [2].

2.2 Feature selection

A set of repeatable image patches is first extracted from the rectified images. These serve as the input for training the random forest. We densely match each

[2] Accelerometers on mobile devices can also provide a vertical VP estimate

patch in an image, searching for the most similar patch across all position and a few discrete scales. We use the distance between the patch templates and their most similar patches in the image as the input feature F for the classifier.

An ideal pool of features D would contain image patches that are both unique as well as repeatable within a class. To obtain such patches, we extract scale invariant DoG keypoints [20] and DAISY descriptors [27] in the rectified images and perform robust feature matching on image pairs in the training set [20, 26]. Outliers are removed using RANSAC and geometric verification after which the correspondences are linked to form multi-view tracks. The set of patches D is then computed by randomly sampling from these tracks [3]. Next, a candidate patch from each track is chosen by selecting the one with the minimum descriptor distance to all other patches in the track. This is added to the set D[4]. For images that did not match any other image, we randomly sample 10 patches corresponding to DoG keypoints in the image. All image patches in our implementation are axis-aligned square patches. They are resampled to 32×32 pixels before computing feature descriptors to be used for dense matching.

We represent the patches using binary (BRIEF) [7] descriptors which allow for very efficient matching. This descriptor is computed by randomly sampling k pixel pairs p_k and p'_k from a 32×32 image patch based on a 2D Gaussian distribution centered on the center pixel, and then setting the k-th bit of the descriptor only if $I(p_k) > I(p'_k)$. Based on experiments, we found k=192 was a good trade-off between accuracy and speed. Distance between descriptors is measured using Hamming distance, which can be computed very efficiently [7].

2.3 Random decision forests

Given a feature vector F computed from densely matching patch descriptors D in the rectified training images, we train a set of random decision trees using standard techniques [6]. The data is recursively split into subsets at each internal node of the binary trees which correspond to binary tests. The leaf nodes of the forest store the class distributions. During classification, the class label is predicted by averaging the class probabilities predicted by all the trees in the forest and selecting the most likely class. We learn binary tests based on single features in F. Thus for each internal node in each tree, we need to learn on which features to make decisions and the corresponding thresholds. The trees are trained independently using standard approaches that randomly selects a subset of potential features from F for training each tree. The best feature at each recursive step is selected as the one which has the largest decrease in Gini impurity. The selection size parameter s (= 20 by default) determines how many random features are considered. No tree pruning is performed in our implementation.

Random forests have several benefits – trees with axis-aligned splits can be stored very efficiently, since only the feature index and a threshold must be stored at each internal node. The classification step is very fast. Storing the features themselves is efficient, since the set of features F^* actually used by the forest

[3] Longer tracks are given preference during random sampling.

[4] In our experiments, D had at most 4000 patches, but more patches can be used too.

is typically smaller than the size of F. Finally, the random selection of features increases robustness to occlusions. We observed that occasionally features are selected on temporary stationary objects in the scene such as parked cars, which will produce non-informative features at classification time. Random selection reduces the effect of such bias in the training sets. By default, we trained our forests with 50 trees where the number of selected features varied between 200 to 3000. Further discussions on the impact of parameters on accuracy versus storage tradeoffs can be found in Section 3.

2.4 Efficient dense matching

Computing the feature vector for the query image prior to classification requires densely matching each patch descriptor in the query image. During training, a subset of features $F^* \subset F$ are selected for the internal nodes of the forest. At query time, we only need to densely match the corresponding descriptors in F^*. However for 500+ descriptors in F^*, brute force Hamming distance computations can be quite expensive. Rectification makes such a dense matching approach more practical, as it restricts the search to 2D translation within multiple scaled version of the image. In practice, we search 10 scales between 0.25X and 1.25X magnification, with a patch size of 32×32 pixels. It is worth noting that no interest points are required by our method.

Significant computational efficiency is obtained using a min-hash approach [11, 10] to discard dissimilar patches. With min-hash, we compute a set of hashes for BRIEF descriptors that have a probability of collision equal to the Jaccard similarity of the two binary vectors. If viewed as sets, the Jaccard similarity of two vectors is the cardinality of their intersection divided by their union. Specifically, the min-hash is the minimum index of a positive bit after the vectors are permuted by a random permutation. As in [11], we compute multiple min hashes and concatenate them into sketches for improved discriminability[5].

We compute a set of sketches for each BRIEF descriptor in F^* and construct an inverse lookup table for them. We scan the image, finding potentially similar patches by detecting sketch collisions using efficient lookup (at least $k = 2$ sketches must be identical). The Hamming distance is computed for these descriptors pairs. In practice, we found that no Hamming distances was computed for 80% of all patches at classification time and on average less than ten distance computations were needed for the remaining patches. Since all the bits of the BRIEF descriptor are not required for computing its min-hash sketches, the bits are computed on demand, as needed by the min-hash function. Thus for 80% of the patches, the full BRIEF descriptor is not computed, saving computation. However computing Hamming distance was usually the bottleneck.

3 Experimental Results

Datasets: We evaluated our approach on the public ZUBUD [24] and CALTECH building datasets [1], each of which has five images of 200 and 50 buildings respectively. We created random sets of query images using a leave one out strategy

[5] In our implementation we use 5 sketches each containing 5 sketches

DATASET	#IMGS	#CLASSES	TRAINING			QUERY STATS	
			#FEATURES	#TREES	DOWNLOAD SIZE	# QUERIES.	ACCURACY
ZUBUD	804	200	4107	50	233 KBYTES	200	92%
CALTECH-50	200	50	2820	50	153 KBYTES	50	90%
SUBURB-48	504	48	2200	60	220 KBYTES	131	50%
TOWN-56	464	56	3138	60	198 KBYTES	62	35.5%

Table 1. Datasets used in our experiments and the performance of the proposed approach in terms of compactness of the classifiers and classification accuracy.

for each building. We also report results on two challenging streetside datasets – SUBURB-48 and TOWN-56, collected by us, where the training images comprise of 504 and 464 images corresponding to an area of about four city blocks. For both these datasets, prominent landmarks (classes) such as buildings, restaurants, stores are labeled manually. Our query set contains 200 images captured by cameras on several mobile devices and have strong viewpoint and appearance changes compared to the training images which were acquired during a different season. Table 1 lists the accuracy of our method and the corresponding download sizes for all four datasets. In Figure 3, we show how varying the selection size and the number of decision trees affect the compactness of our classifiers. The accuracy improves as these parameters are increased but starts to converge for a selection size of about 20 and with 50 decision trees.

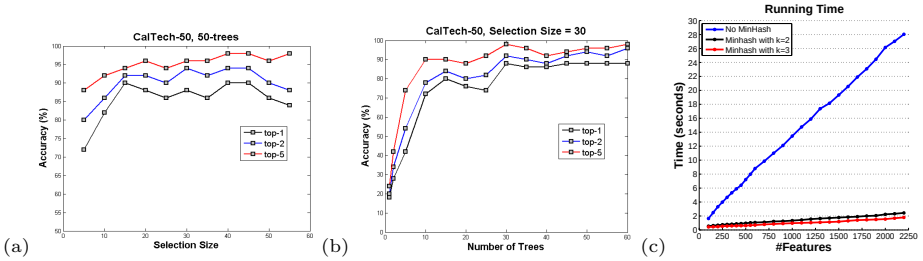

Fig. 3. Accuracy on CALTECH-50 with different RDF parameters – (a) selection size and (b) the number of trees. Accuracy is the fraction of correctly classified queries considering top k results (k = 1, 2 and 5). (c) Feature extraction timings: Brute force distance computation timings shown in blue. Timings for our method is shown in red/black, where Hamming distance is computed only when at least k min-hash sketches collide.

Comparison with BoW and SIFT: To assess the suitability of our approach for a mobile device, we compare the accuracy and download size tradeoff of our method with SIFT matching [20] and BOW method of [21]. These are considered as state of the art for location recognition when memory footprint and storage is not an issue. The classification accuracy is the percentage of query images correctly recognized. For SIFT, the retrieved images for each query are ranked by the number of matches. The ranked image list is mapped to a list of

Fig. 4. Four example queries on TOWN-56 and SUBURB-48 datasets. The original and rectified images are shown on the left and the recognized building is shown on the right.

classes by finding the first occurrence of each class in the sorted list. To force memory/download constraints on SIFT matching, only a fraction of SIFT keypoints were sampled from the database image and used for matching. Further the descriptor vectors were quantized to k-bits entries (k = 1 to 8). Similarly storage constraints were forced on the BoW method by choosing vocabularies with 1K to 100K words and quantizing the histogram entries to use 1 to 16 bits. BOW histogram were represented as sparse vectors. Figure 5 shows that both SIFT and BOW accuracy degrades significantly when the memory/storage size is lowered. In our method, the download size is varied by choosing different RDF parameters. Rectified input images were used in all the comparisons.

Our method outperforms both BoW and SIFT in accuracy even though our classifiers are one or more orders of magnitude more compact. For example, on TOWN-56 and SUBURB-48 datasets our method had an accuracy of 36% and 49.5% respectively with about 200KB storage size. While SIFT matching did not work at all, BOW methods had an accuracy of approximately 8% and 12% for the two datasets. The best performance with BOW and SIFT was obtained with 2MB+ and 40MB+ storage size in both datasets. The results on CALTECHR-50 is similar; our method has an accuracy of 90% with 100KB of storage whereas BOW methods had an accuracy of 50% when compressed to about 200KB.

Download size. Each internal node of our decision trees require roughly 5.5 bytes to represent a feature index, an integer threshold and two pointers. The total download size can be approximated as $24N + 6.5TH$ bytes (assuming we have fewer than 1024 classes), where the random forest selects N patches (each of which is represented using 32 bytes). T is the number of trees in the RDF and H is the average tree height. For C classes, we need decision trees with height $\log_2(|C|)$. With hundreds of classes, the storage size is typically dominated by the feature descriptors which are selected during the training stage.

Running Time. For dense matching, we resize the images setting its larger dimension to 512 pixels. When searching 10 levels of scale from 0.25X to 1.25X, the running time varies between 0.5s to 1.5s for most datasets. The time complexity is linear in the number of features. Figure 3(c) shows the running time for performing dense matching on an image of resolution 512×420 on a laptop with a single core 2.66GHz processor. For the min-hash based approach, we computed Hamming distance between descriptors only when k out of five sketches matched between a pair of descriptors. Figure 3(c) shows running time for $k = 2$ and 3 and shows how the min-hash produces an order of magnitude speedup.

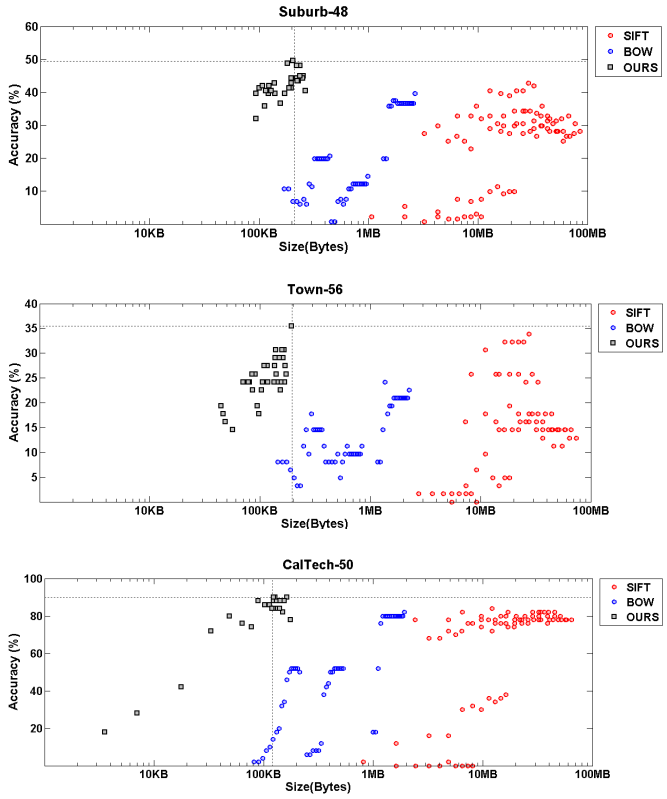

Fig. 5. Classification Accuracy: On both streetside datasets and Caltech-50, our method outperforms BoW and SIFT even though our classifiers are 1 or 2 orders of magnitude more compact. Each method was configured with different storage/download sizes to obtain the scatter plots. The X-axes are in log-scale. The best configuration of our method is shown using dotted lines. Our accuracies are reasonable under 100KB whereas BOW performs poorly under 100KB and SIFT matching completely fails.

4 Conclusion

We have proposed a new discriminative method for classifying urban landmarks that exploits geospatial scope to train very compact classifiers with efficient query processing capabilities. Our method is currently less robust to recognizing landmarks across different seasons. In the future we will focus on improving the recognition accuracies using diverse multi-season imagery for training.

References

1. M. Aly, P. Welinder, M. Munich, and P. Perona. Towards automated large scale discovery of image families. *CVPR Workshop on Internet Vision*, pages 9–16, 2009.
2. Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9, 1997.
3. C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *ISMAR*, pages 73–82, 2009.

4. G. Baatz, K. Koser, R. Grzeszczuk, and M. Pollefeys. Handling urban location recognition as a 2d homothetic problem. In *ECCV*, 2010.
5. M. Bansal, H. S. Sawhney, H. Cheng, and K. Daniilidis. Geo-localization of street views with aerial image databases. In *MM'11*, pages 1125–1128, 2011.
6. L. Breiman. Random forests. *Machine Learning*, 45, 2001.
7. M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *ECCV (4)*, pages 778–792, 2010.
8. V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: Compressed histogram of gradients a low bit-rate feature descriptor. In *CVPR*, pages 2504–2511, 2009.
9. D. Chen, G. Baatz, Köser, S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *CVPR*, 2011.
10. O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, 2009.
11. O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, 2008.
12. J. Hays and A. Efros. Im2gps: estimating geographic information from a single image. In *CVPR*, 2008.
13. A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, pages 2599–2606, 2009.
14. H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311, 2010.
15. J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. In *ECCV*, 2010.
16. V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *PAMI*, 28:1465–1479, September 2006.
17. 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.
18. Y. Li, D. Crandall, and D. Huttenlocher. Landmark classification in large-scale image collections. In *ICCV*, 2009.
19. Y. Li, N. Snavely, and D. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010.
20. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. of Computer Vision*, 60, 2004.
21. D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168, 2006.
22. D. Robertson and R. Cipolla. An image based system for urban navigation. In *BMVC*, pages 819–828, 2004.
23. G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007.
24. H. Shao, T. Svoboda, and L. V. Gool. Zubud-zurich buildings database for image based recognition. Technical report, No. 260, ETH Zurich, 2003.
25. J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.
26. P. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV WS-LAVD*, 2009.
27. S. Winder, G. Hua, and M. Brown. Picking the best DAISY. In *CVPR*, pages 178–185, 2009.
28. A. Zamir and M. Shah. Accurate image localization based on google maps street view. In *ECCV*, 2010.
29. W. Zhang and J. Kosecka. Hierarchical building recognition. *Image Vision Comput.*, 25(5):704–716, 2007.