Chapter 15
A Memory Efficient Discriminative Approach for Location-Aided Recognition

Sudipta N. Sinha, Varsha Hedau, C. Lawrence Zitnick and Richard Szeliski

Abstract In this chapter, we describe a visual recognition technique for fast recognition of urban landmarks on a GPS-enabled mobile device. Most existing methods offload their computation to a server by uploading the query image. Over a slow network, this can cause a latency of several seconds. In contrast, our approach requires uploading only the approximate GPS location to a server after which a compact, location-specific classifier is downloaded to the device and all subsequent computation is performed on it. Our approach is supervised and involves training compact random forest classifiers (RDF) on a database of geo-tagged images. The feature vector for the RDF is computed by densely searching the image for the presence of selective discriminative local image patches extracted from the training images. The images are rectified using detected vanishing points and binary descriptors allow for an efficient search for the discriminative patches, a step that is further accelerated using min-hash. We have evaluated the performance of our approach on representative urban datasets where it outperforms traditional methods based on bag-of-visual-words features or direct matching of local feature descriptors, neither of which are feasible approaches when processing must occur on a low-power mobile device.

S.N. Sinha (✉)
Microsoft Research, Redmond, WA, USA
e-mail: sudipsin@microsoft.com

V. Hedau
Apple, Cupertino, CA, USA
e-mail: varsha.hedau@gmail.com

C.L. Zitnick
Facebook AI Research, Palo Alto, CA, USA
e-mail: zitnick@fb.com

R. Szeliski
Facebook, Seattle, WA, USA
e-mail: szeliski@fb.com

A.R. Zamir et al. (eds.), *Large-Scale Visual Geo-Localization*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-25781-5_15

15.1 Introduction

The ubiquity of cameras on GPS-enabled mobile devices such as smartphones provides great utility for determining the location and contents of an image. It may be used to learn more about a specific landmark turning the phone into a device for visual queries or for automatic image-tagging allowing the images to be searched and organized later on. We address the problem of recognizing the landmark or location from a single image where a predefined set of locations and landmarks is available. In this chapter, we will describe a technique where the query processing can be implemented to run completely on a low-power mobile device.

The feasibility of location recognition has recently increased with the availability of large databases of dense geo-tagged imagery, e.g., Flickr photo collections and streetside imagery [24, 25] for cities. Recently, several approaches to location recognition have been proposed based on such databases [6, 15, 23, 31–33, 46, 53].

The main challenge in landmark or location recognition arises from variations in scale and scene appearance due to a diversity of camera viewpoints or due to illumination changes from time-of-day, weather, seasons etc. Although the intrinsic scene appearance does not change a lot, the visibility of landmarks covering a wide area can change dramatically with viewpoints. In addition, foreground objects such as people or cars can clutter the scene at busy urban locations. Finally, the appearance of a location may change temporally due to seasonal variation, or permanently due to construction, new store ownership, new billboards, etc.

Broadly speaking, most existing methods for landmark classification pose it as an image retrieval task [23, 43]. which requires ranking the database of geo-tagged images based on similarity to the query image. The similarity score is typically computed by comparing sets of local feature descriptors extracted at scale-invariant keypoints detected in the images. Although fast retrieval techniques exist for databases with millions of images, constant access to this huge database is necessary with consequent high storage and memory costs. For high precision, an expensive reranking step is often used for geometric postverification of feature matches over image pairs. Recently proposed alternatives to the retrieval approaches include direct image matching to sparse precomputed structure from motion (SfM) point clouds [33, 34, 36, 45]. Despite the higher compression in these methods, their storage costs and computational requirements are still quite significant.

Image categorization methods, on the other hand, use labeled data to train classifiers that are typically efficient to evaluate at query time and have lower storage costs [8, 32]. However, most of the existing techniques have traditionally focused on objects or scene categories. Large-scale and fine-grained classification has recently gained interest among researchers, however, with the exception of [8, 21, 32], landmark or location classification has received less attention from a purely classification or categorization standpoint.

15.1.1 Overview

With the advent of GPS-enabled modern mobile devices, one important and new aspect of location or landmark recognition is problem scoping, i.e., limiting the set of locations or landmarks that need to be searched. This is because GPS devices can easily narrow down a user's location to within approximately a hundred meters. In this chapter, we describe a memory-efficient discriminative approach [32] to location recognition that exploits approximate GPS coordinates. Instead of storing feature vectors for all the geo-tagged images in a database, we store compact, location-specific classifiers. Each classifier for a certain landmark is trained to distinguish it from only the nearby landmarks that lie in close proximity within the coarse location predicted using GPS. Our location classifiers are based on Random Decision Forests (RDF) [9] trained to predict the correct location or landmark within scope based on a global feature descriptor computed from the query image. Each dimension of this feature vector encodes the presence or absence of a specific discriminative local image patch within the query image.

During training, we identify potential candidates for such discriminative patches from the images of all the locations under consideration. Next, each candidate in this pool is densely matched to patches in all the training images and the matching score of the most similar patch found in the image is recorded in the global feature descriptor. All images are rectified prior to dense matching [6, 38, 44] to reduce the search space to just scale and 2D position and for improving robustness to perspective distortions. At query time, we propose to use a min-hash approach to speedup the feature vector computation. Specifically, this involves using approximate nearest neighbor search to identify the most similar patch within the query image, for each of the discriminative patches used by the RDF classifier.

We have evaluated our method on two public benchmarks for landmark recognition and two sets of urban images that we captured with a smartphone in a typical urban streetside scene. In the latter case, the query images were collected almost a year after the time the training imagery was acquired. Notable appearance changes were observed in the query images due to seasonal changes, viewpoints as well as structural changes due to construction. We show that even when approximately a hundred locations or landmarks must be discriminated from each other, our proposed method uses only about 120 KB for storage without sacrificing much accuracy and its performance is comparable to some of the most accurate methods that use significantly more storage. Thus, the approach appears to have several advantages. First, the RDF classifier achieves high accuracy despite the compactness constraints. Second, the min-hash-based dense matching and efficient feature vector computation step makes the query time computation feasible on a low-power mobile device. Image rectification helps to reduce the search space for dense matching and provides tolerance to large viewpoint changes that induce large perspective distortions in the image. Finally, RDF classifiers also implicity perform feature selection and this increases the overall likelihood of choosing more repeatable and temporally stable local dictionary features without additional postprocessing during the training stage.

15.1.2 Related Work

A number of location or landmark recognition approaches utilize local keypoint feature descriptors but improve their matching accuracy by employing offline learning. Knopp et al. [29] remove confusing features in a bag-of-words model, Torii et al. [49] exploit repeating features whereas Li et al. [33] prioritize the matching of repeatable and frequently occurring features. Turcot and Lowe [50] remove features with low repeatability and utilize features co-occuring in neighboring images. A tree-based descriptor quantization approach that maximizes the information gain of quantized visual words was proposed by Schindler et al. [46]. Zamir et al. [53] remove noisy matches between images using a variant of the descriptor distance ratio test [37] and Zhang et al. [54] uses a motion estimation technique that is robust to outliers. Our approach uses keypoint detection and local descriptor matching to identify a pool of candidates used for constructing the dictionary of discriminative patches. However, the global image descriptor or feature vector used by the RDF classifier is then computed using efficient min-hash-based dense matching techniques within the whole image rather than only at detected keypoints. Min hash, which is a common locality sensitive hashing technique, has been shown to be effective for efficient near-duplicate image search in large databases [41, 42].

The significance of image rectification using vanishing points as a preprocessing step for wide-baseline image matching and location recognition has been studied [6, 44]. This step provides higher invariance to perspective distortions when matching rectilinear structures in images of building facades in street scenes [38]. A heading dependent bag-of-features representation for panoramic images has been proposed by Guan et al. [22]. Efficient feature extraction and compactness can be obtained using compressed sensing techniques [22] or with low bit-rate compressed visual feature descriptors [14]. Another alternative approach to obtaining compact discriminative descriptors involves learning discriminative embeddings [26] or mapping the high-dimensional descriptors into the Hamming space to produce short binary codes that can be compared very efficiently [10].

For recognizing images of urban scenes, a number of recent methods employ offline structure from motion computation as a preprocessing step [17, 27, 35, 36, 39, 45]. Benchmarks for mobile image-based localization have been constructed to evaluate the performance of recognition systems [15, 16]. Clemens et al. [4] proposed such a method for localization on a mobile phone. Their approach is computationally efficient but the memory requirements grows linearly with scene size. Arth et al. [3] propose a similar technique that uses a panoramic image as a query and exploits inertial sensors available on most mobile devices. A number of recent augmented reality techniques related to image-based localization are discussed in [5].

For real-time image-based localization on images or video, efficient search index schemes have been used for matching image patch descriptors [36, 45]. Scale-invariant feature extraction at query-time can be slow but it can sometimes be avoided by increasing the storage redundancy [36]. For real-time performance on mobile devices, a part of the computation is often offloaded to a server [39]. Discrimina-

tive indexing strategies have been proposed recently for improving the accuracy of descriptor-based matching and classification of landmarks [12, 13]. Specifically, Cao and Snavely [12] exploits the implicit graph connectivity between known locations of the training images to learn more accurate image descriptors [13] leading to compact model representations without sacrificing area coverage or discriminative information.

Generic scene recognition can be achieved with approximate nearest neighbor search based on high-dimensional features consisting of color and texture histograms, line features, and global descriptors that encode image appearance, such as GIST [23]. Image-based features have also been shown to be effective for recognizing natural scenes and mountainous terrain [7]. In certain domains, hierarchical classification approaches [54] have been explored. However, as mentioned earlier, these methods all have high memory requirements as the feature vectors of the training set must be stored and accessed at query time.

Our work follows a different line of work that is more closely related to [8, 19, 21, 32]. Unlike localization methods, these techniques are not designed to estimate a metric camera pose or to register the image to a 3D scene model. Nevertheless, the predicted location label is sufficient for answering visual queries or automatically generating tags or useful metadata for the query images. Doersch et al. [19] propose a method to automatically discover discriminative mid-level visual elements from large-scale streetside imagery in urban scenes. These visual elements are shown to be extremely useful for geo-localization. The three methods [8, 21, 32] formulate location recognition as a multiclass classification problem and train compact linear classifiers for predicting the image label. Specifically, these methods train different variants of support vector machines (SVM) on high-dimensional global image feature vectors that are computed using a visual codebook and popular feature encoding techniques. Bergamo et al. [8] propose a method to learn a highly discriminative codebook by leveraging image correspondences recovered from Internet image collections using a modern structure from motion (SfM) technique. For large photo collections of landmarks and tourist scenes, their approach is an efficient alternative to [19] for discovering discriminative and repeatable visual features in the scene.

In contrast to [8, 21, 32], we use random forests for location classification. Random decision forests (RDF) [9] are applicable to a wide variety of tasks in computer vision, such as multiclass classification, regression, and density estimation as discussed in [18]. Some of its early applications include shape recognition [2], semantic segmentation [28], and efficient keypoint matching [30]. This chapter describes how they are also well suited for training memory-efficient location classifiers that can be compactly represented. Typically, RDFs have two advantages over SVMs. First, they extend naturally to multiclass classification tasks. Second, feature selection which is crucial for a compact representation, is performed implicity when training RDFs but may require expensive postprocessing when training SVMs [51].

15.2 Learning Location Classifiers

In this section, we describe our method for learning location classifiers using a RDF classifier. We assume that a training set of geo-referenced images has been collected within an area that can be predicted directly using GPS. We assume that the images of all the important locations or landmarks within this area in the training set have been labeled. Each location or landmark is treated as a different class in our method.

Classification is performed using a dictionary of highly discriminative image patches that are extracted from the geo-referenced images using a keypoint detector during offline training. The input to the RDF classifier is a high-dimensional image feature vector whose dimensionality is equal to the number of selected patches in the dictionary. Each dimension of the feature vector stores the matching cost between a specific dictionary patch and the patch in the input image that is most similar to it. The use of dense matching typically decreases the variance of the input features to the RDF compared to when sparse keypoint features are used [32]. However, to efficiently identify a set of discriminative dictionary patches, we first perform pairwise image matching [37] on the set of training images of each landmark or location and extract a subset of the most repeatable patches within these images. These typically correspond to the keypoints whose feature descriptors were matched reliably. The collection of such patches extracted from the different location classes serves as a pool of candidate features for the subsequent classifier training stage.

An overview of our location recognition system is illustrated in Fig. 15.1. The various stages of this system are described in the following sections. First, in Sect. 15.2.1, we describe our approach for rectifying the image using the extracted vanishing

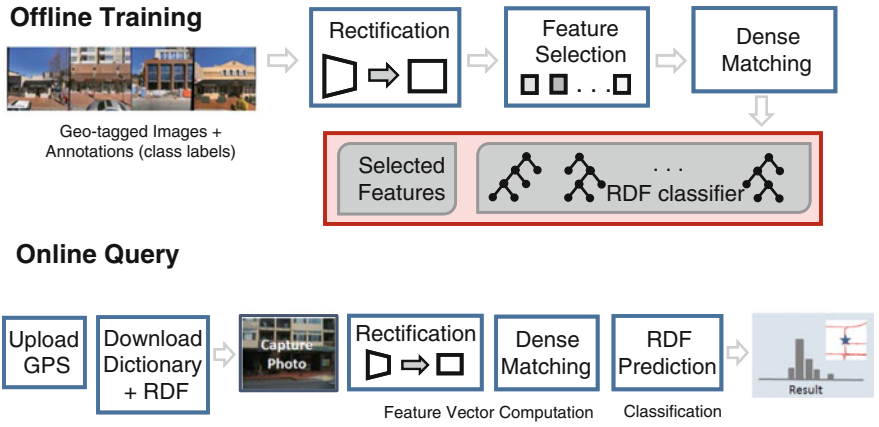

Fig. 15.1 An overview of our classification approach to location recognition. The training stage involves image rectification followed by the selection of discriminative local patches via image matching. Dense matching and search for each discriminative patch is used to construct feature vectors for a random decision forest (RDF) classifier. On a mobile device, at query time, only the GPS information needs to be uploaded to the server after which a compact, location-specific RDF classifier can be downloaded to the device for local computation and processing

points. Next, in Sect. 15.2.2, our approach for selecting a pool of candidate discriminative patches to be used as a dictionary for the classifier is described. Finally, Sect. 15.2.3 describes the details of training the random forests. Our method for efficiently constructing the feature vector using approximate nearest neighbor patch search is described in Sect. 15.2.4.

15.2.1 Image Rectification

Planar objects such as building facades may undergo severe perspective distortion in the image depending on the orientation and position of the camera. Searching over the full eight degrees of freedom provided by perspective distortions is computationally prohibitive for dense matching. If the focal length of the camera is known and vanishing points can be identified, the degrees of freedom can be reduced to three in a process called image rectification. Dense matching can then be performed by searching only in position and scale. Previous works have also used rectification to increase the repeatability and discriminability of interest points [6, 44].

We perform metric rectification by automatically detecting orthogonal vanishing points and using an approximate estimate of the focal length. We also assume that query images will have low camera roll (camera is mostly upright), thereby allowing us to identify the vertical vanishing point. As in *upright* SIFT [6], the absence of rotational invariance makes our features more discriminative.

We detect vanishing points in two stages. First, the vertical vanishing point in the image is estimated via RANSAC [20] on 2D line segments subtending a small angle to the vertical. For speed and accuracy, longer lines are sampled more frequently during the randomized hypothesis generation step. When the focal length is known, the vertical vanishing point determines the position of the horizon in the image. Horizontal vanishing points are found using a 1-line RANSAC on nonvertical lines that intersect the horizon. In the case of an unknown focal length, the 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 reliable estimates of two orthogonal vanishing points are recovered, we rectify the image using the 2D homography, $H = \mathrm{K}\mathrm{R}^{-1}K^{-1}$ where, the matrix $K = \mathrm{diag}([f\,f\,1])$ represents camera intrinsics with normalized focal length f and R represents the 3D camera rotation in the frame of the orthonormal vanishing directions. The rectified image is cropped whenever the local distortion caused by the perspective warp due to H exceeds a maximum threshold. Several examples of rectified image are shown in Fig. 15.2. An accurate metric rectification is desirable but not essential for our approach to work. When only the vertical vanishing point is detected, we perform roll correction, thereby eliminating one degree of freedom in camera rotation. Mobile devices are nowadays equipped with accelerometers which provide an approximate estimate of the vertical direction. When gyroscopes and magnetometers are available, they may provide alternative approaches to rectifying the image based on training data.

Fig. 15.2 Example query images from a mobile phone camera. The *top row* shows the original images, while the rectified versions of these images computed using our method are shown the *second row*

15.2.2 Selecting Discriminative Patches

In this section, we describe our approach to select a pool of discriminative image patches across all the location classes in the dataset under consideration. Let D denote the total number of candidate patches after resampling the original patches to 32×32 pixel each. These patches will be used to construct the D-dimensional feature vectors for training the Random Decision Forest (RDF).

To compute the k-th dimension of the feature vector, a dense search is performed within an image to find the patch most similar to the k-th patch in the candidate pool. This search is performed across all positions and a few discrete scales. For computational efficiency, we use BRIEF descriptors [11] to represent patches and compare their visual similarity. The BRIEF descriptor is a binary descriptor that is computed by randomly sampling m pixel pairs $\{(p_j, q_j)\}_{j=1}^{m}$ from the underlying 32×32 patch based on a 2D Gaussian distribution centered on the patch center. The j-th bit of the BRIEF descriptor is set to 1 when the intensity at pixel p_j is greater than the intensity at pixel q_j. Based on our experiments, we found $m = 192$ to provide a good compromise between accuracy and speed. The similarity between two patches is measured by the Hamming distance between their corresponding BRIEF descriptors. This can be computed very efficiently; on certain modern processors a dedicated *popcount* instruction exists for performing the computation in a single operation [11]. Finally, the k-th dimension of the feature vector is assigned to the minimum Hamming distance between the descriptor of the dictionary candidate and all descriptors extracted from the image.

An ideal set of candidate patches are ones that are both unique as well as repeatable within a location class. To select such patches, we first extract scale-invariant DoG [37] keypoints and DAISY descriptors [48, 52] from each image in the training set. Next, pairwise image matching is performed between all pairs of images from each

location. The initial putative matching is done using approximate nearest neighbor search on the DAISY descriptors based on a standard kd-tree index [37]. The putative matches are then geometrically verified using RANSAC with an epipolar geometry model to prune outliers from the putative matching stage. The subset of two-view matches that satisfy the respective epipolar geometries are subsequently linked to form a track. First, a small number of tracks are randomly selected. The random sampling at this stage is biased toward favoring longer tracks over shorter tracks. Once a track has been selected, the specific patch within the set of patches in the track whose descriptor has the minimum distance to all the other descriptors in the track is chosen as the representative patch.

This process is repeated for all the location classes to select a uniform number of candidate patches from all the different locations. In our experiments we specify a budget of 4000 features uniformly divided across all the classes. However, for smaller datasets involving fewer classes, we expect that a lower number of candidates will be sufficient. For locations for which epipolar matching failed to produce long tracks on the training images, we select patches corresponding to DoG keypoints at random positions and scales from randomly chosen images and add them to candidate pool. All image patches are axis-aligned square patches and are resampled to 32×32 pixels before BRIEF descriptors are computed.

15.2.3 Random Decision Forests

After densely matching the D dictionary patches to all the images in the training set, the feature vectors required to train the RDF classifier are now available. The corresponding location labels for the training images are also available. In practice, these labels could be obtained from Flickr image-tags, by manually adding metadata to streetside imagery or by clustering the GPS coordinates of geo-tagged images in the training data.

Our random forest classifier consists of multiple binary decision trees that are trained independently. We use bagging [9] to create random subsets of the training data for training each decision tree. In our system, we randomly select 100 % of the training data with replacement, i.e., selecting n samples from the n original training samples with replacement which generates subsets with 63.2 % expected unique samples from the training set, the rest being duplicates.

Each node in a decision tree is optimized using the randomized greedy approach described in [18]. A fixed number of weak learners are randomly sampled. We refer to this hyperparameter as the *selection size*. Each of these weak learners correspond to a feature space partition along a randomly chosen dimension with the split occurring at a randomly chosen value. The final hyperplane at each node is found by selecting the split that resulted in the highest information gain which requires calculating the entropy of the empirical distributions before and after each split. Each axis-aligned weak learner can be represented compactly since only an index for the feature dimension and a threshold needs to stored. The leaf nodes of each tree store the

resulting class distributions. During classification, the final result is obtained by averaging the probabilities of the classes predicted by the different trees in the forest. Unlike other implementations of random forests, no tree pruning is performed.

Note that the decision trees are already quite compact due to the choice of the underlying weak learners. However, another level of compactness is possible due to the following fact. In general, certain dimensions in the D-dimensional feature vector are never included in any of the weak learners used by the trained RDF. Let D^* denote the number of dictionary elements that are actually utilized in the RDF. This is then the effective dimensionality of the dictionary required at query time and dimensionality of the feature vector needed for evaluating the classifier at query time. In our experiments, we train forests with 50 trees and D^* typically ranges from 200 to 3000 depending on the number of classes in the dataset. The greedy approach for training random forests are controlled by the selection size hyperparameter which injects randomness and ensures that a wider variety of features are selected by the forest and this is known to encourage generalization. For example, the training images for a particular location may have a parked car seen in all the images. Even though the patches corresponding to the parked car may be the most discriminative given the training data, randomizing the node optimization of the random forest provides better generalization for the future when the scene appearance may have changed significantly.

15.2.4 Efficient Dense Matching

Computing the D^*-dimensional feature vector is the main computational bottleneck at query time. We now describe an efficient solution which is based on using min-hash for approximate nearest neighbor matching [41, 42]. Min-hash is used to compute a set of hashes for binary vectors that have a probability of collision or matching equal to the Jaccard similarity of the two vectors. When viewed as sets, the Jaccard similarity of the two vectors is just the cardinality of the intersection of the two sets divided by the set union. The min-hash of a binary string is the smallest index at which a bit is set (to 1) after a random permutation is applied to the original vector. Min-hash is appropriate for detecting exact duplicates very efficiently but can have low recall. To boost the recall, the selectivity of the hashes can be increased by concatenating multiple min-hashes into a *sketch*.

Let Q denote the set of BRIEF descriptors for the dictionary elements in our system. Let P denote all the BRIEF descriptors for patches densely sampled in the image. We compute t different sketches for each BRIEF descriptor, each of which consists of a concatenation of r min-hashes. A pair of descriptors (p_i, q_i) such that $p_i \in P$ and $q_i \in Q$ are deemed a potential match when s out of t sketches are found to be identical. Full Hamming distances are computed using the original descriptors only for the subset of descriptor pairs that are deemed potential matches. In our implementation, we set $t = 5$, $r = 5$, and $s = 2$, respectively.

Before the min-hash-based method can be used efficiently, a mechanism for efficient retrieval of descriptor sketches is required. This requires a small amount of precomputation on the mobile device after the dictionary is downloaded for the first time. The sketches for the set Q are first extracted and then an inverted index is constructed over those sketches to allow efficient retrieval of the descriptors in Q given any query sketch later on.

The actual dense matching is then performed using a brute-force linear scan. As described earlier, the min-hash technique is used to skip comparisons that are unlikely to yield the nearest neighbor in Hamming space. In practice, the set of descriptors for which the Hamming distance is computed is quite small and this is key to the speedup in the dense matching stage. On average, we found that Hamming distance calculations were skipped for 70–80 % of all patches extracted from query images.

Since all the bits of the BRIEF descriptor are not required for computing its min-hash sketch, another optimization is possible during dense matching. This involves computing the bits of the BRIEF descriptor on demand, i.e., only when needed by the min-hash function. Thus, for a majority of the image patches which get rejected using the min-hash technique, the full 192-dimensional BRIEF descriptor is never computed, rather partial descriptors are computed where only the bits essential to compute the min-hash sketches are evaluated.

Despite the various optimizations, in our experiments, we found that Hamming distance computation for potential matching pairs was often the bottleneck especially when the dictionary size grows larger (100 or above). However, Hamming distance can be computed very fast on certain processors [11] and such hardware acceleration is expected to be available on mobile platforms in the future making our approach feasible.

15.3 Experimental Results

To assess the feasibility of our approach for a mobile device, we analyze the download size of our classifiers and compare its accuracy to a method based on bag-of-visual words (BoW) as well as to a direct feature matching method. These results are shown in Fig. 15.4.

15.3.1 Datasets

We perform our evaluation on two public landmark datasets—ZuBuD [47] and CALTECH building datasets [1], each of which has five images of 200 and 50 buildings, respectively. For these benchmarks we created random query image sets using a leave one out strategy on each class. We also report results on two challenging streetside datasets—SUBURB-48 and TOWN-56, collected by us, where the training

Table 15.1 Summary of datasets used in our experiments and the performance of our proposed approach

Dataset	#Imgs	#Classes	Training			Query stats	
			#Features	#Trees	Download size (KB)	#Queries	Accuracy (%)
ZuBuD	804	200	4107	50	233	200	92
CalTech-50	200	50	2820	50	153	50	90
Suburb-48	504	48	2200	60	220	131	50
Town-56	464	56	3138	60	198	62	35.5

The number of features refers to the dictionary size and the download size refers to the storage used for the dictionary as well as the classifier

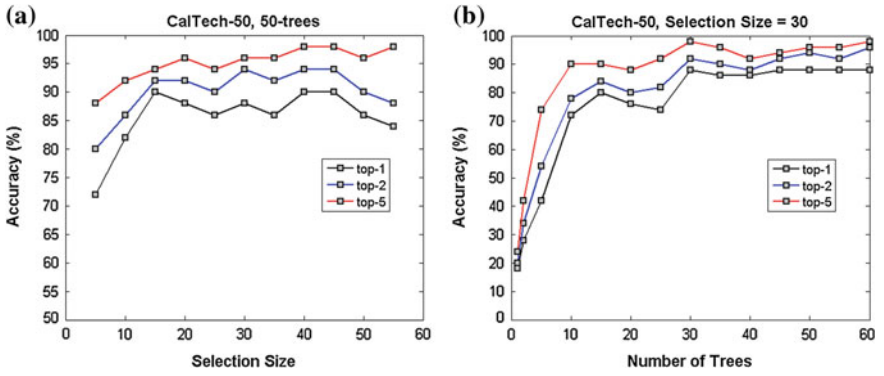

Fig. 15.3 The RDF's classification accuracy depends on the two hyperparameters—selection size and the number of decision trees in the forest as shown for the experiments on the CALTECH-50 building dataset. The top-k classification accuracy is shown here for $k = 1$, 2, and 5, respectively. **a** The selection size parameter controls the degree of randomness—increasing the value of this parameter first improves accuracy but using larger values hurts generalization. **b** Increasing the number of trees also consistently improves accuracy but also increases the storage cost. The accuracy remains almost the same beyond 40–50 trees. Similar trends are observed on the other datasets

images comprised of 504 and 464 images each of an area the size of four city blocks. For both datasets, prominent landmarks (classes) such as buildings, restaurants, and store fronts were manually identified and labeled. 200 images captured by pedestrians with mobile phones were used for queries. The database images were captured in a different season from when the query images were captured. Significant appearance variations caused by changing seasons and weather, and illumination and viewpoint variations make these datasets challenging for recognition (Table 15.1).

Our experiments demonstrate that compact classifiers and dictionaries can be constructed for tasks involving up to 200 locations or landmarks without sacrificing accuracy. Figure 15.3 shows the effect of RDF hyperparameters on the classification accuracy. The selection size parameter and number of decision trees affect the compactness of our classifiers. The accuracy improves as these parameters are

increased but starts to converge after a selection size of about 20 and with 50 decision trees in the forest.

15.3.2 Comparison with traditional methods

We compare our method with SIFT feature matching [37] and BoW [40] methods, with different configurations that result in different storage size required by the different methods. 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 classes by finding the first occurrence of each class in the sorted list. To impose 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 = 1–8). In a similar fashion, storage constraints were placed on the BoW method by choosing vocabularies with 1000–100,000 words and quantizing the histogram entries to use 1 to 16 bits. BoW histogram were represented as sparse vectors. Figure 15.4 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. Note all comparisons were performed on rectified images computed based on automatic vanishing point detection.

For all three datasets on which the comparison was performed, 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 200 KB 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 2 MB+ and 40 MB+ storage size in both datasets. The comparison on CALTECH-50 is similar with the accuracy of our method being about 90 % with 100 KB of storage whereas BoW methods had an accuracy of about 50 % when compressed down to about 200 KB (Figs. 15.5 and 15.6).

This produces probability distributions at the leaf nodes which can be represented with a single integer, since all the training examples at a leaf node belong to the same class. Thus, with C classes, we need $\log_2(|C|)$ bits of storage at each leaf node. Each internal node requires 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. With more than a hundred classes, the storage size is typically dominated by the selected feature sets in our method.

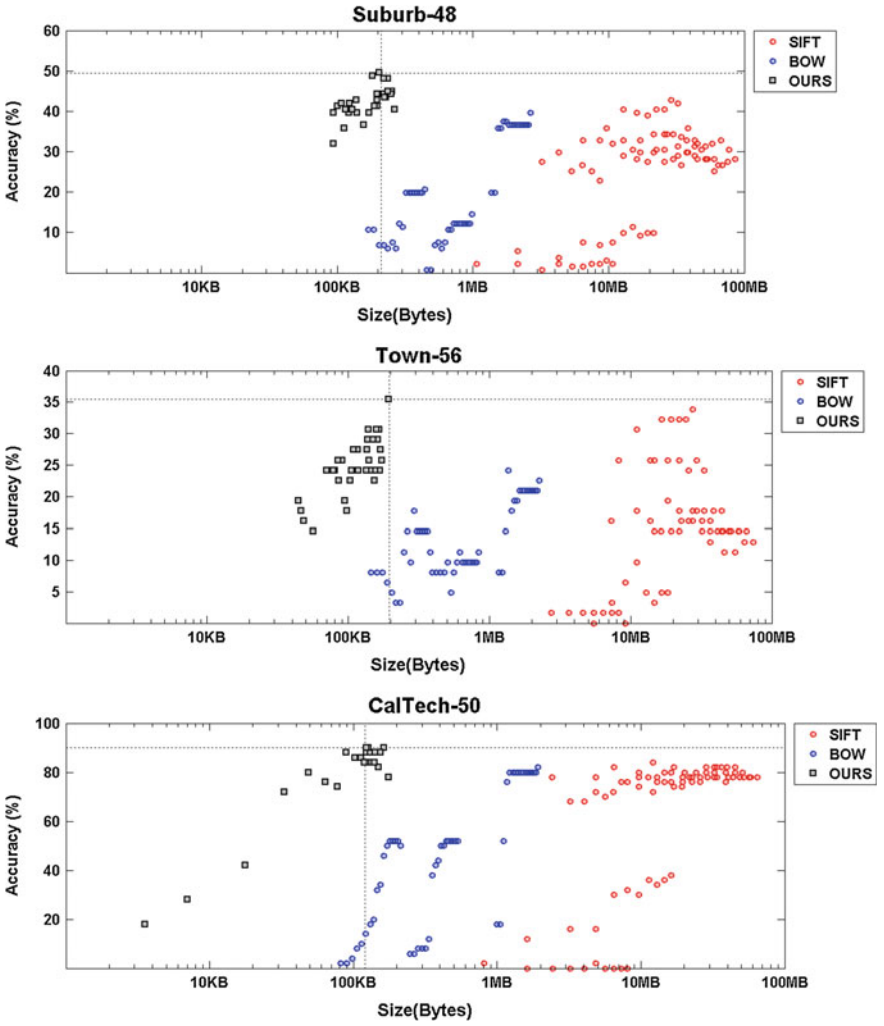

Fig. 15.4 COMPARISONS: This figure compares the classification accuracy of our method with that a BoW and a direct SIFT matching method which are considered state of the art for location recognition when memory footprint and storage is not an issue. For the two streetside datasets as well as the public CALTECH-50 dataset, our method outperforms both BoW and SIFT in accuracy even though our classifiers are one or more orders of magnitude more compact. Each method was configured to run with different storage/download sizes which generated the scatter plots shown here. The most accurate configuration for our method is indicated using dotted lines. Our method works reasonably well under 100 KB whereas BoW perform very poorly under such extreme compression and SIFT does not work at all. The X-axis in all plots is in log-scale

Fig. 15.5 Examples of successful location recognition queries using our method on the TOWN-56 and SUBURB-48 datasets. For each figure, the image in the *left column* and *top row* is the query image captured using a mobile device and the corresponding rectified image is shown in the *left column* and *bottom row*. The four images in the *middle* and *right columns* for all examples show the database images of the building which ranked highest among all the landmark or building classes for the particular dataset in question

15.3.3 Running Time

For the dense matching step, we resize the image such that its larger dimension is 512 pixels. When searching ten discrete levels of scale between a magnification factor of

Fig. 15.6 Examples of successful location recognition queries using our method on the TOWN-56 and SUBURB- 48 datasets. For each figure, the image in the *left column* and *top row* is the query image captured using a mobile device and the corresponding rectified image is shown in the *left column* and *bottom row*. The four images in the *middle* and *right columns* for all examples show the database images of the building which ranked highest among all the landmark or building classes for the particular dataset in question

$0.25\times$–$1.25\times$ of the size of the resized image, the running time for query processing varies between 0.5 and 2.0 s on a single CPU core for all our datasets. The time complexity of the dense matching step which dominates query processing is linear

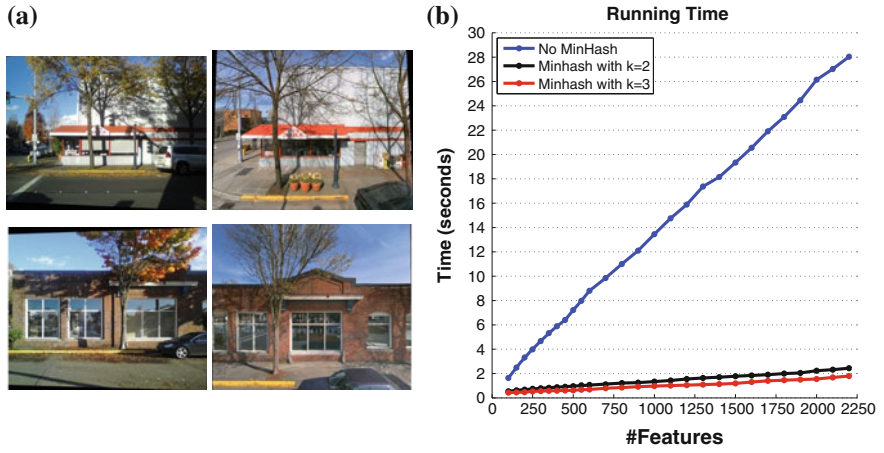

Fig. 15.7 **a** A couple of failure examples are shown (*query images on the left*). The database images shown on the *right contains shadows* and *parked cars* and was captured in a different season. **b** Running times for the dense matching step are reported. The *blue curve* shows timings when brute-force Hamming distance computation is performed during dense matching. The *red* and *black curves* shows timings for our min-hash-based approach, where Hamming distance is computed only when at least k min-hash sketches of the binary descriptors are identical

in the dimensionality of the feature vector, i.e., the number of patches in the dictionary. Figure 15.7b shows the running time for performing dense matching on a 512×420 pixel image on a laptop with a single core 2.66 GHz processor. For the min-hash-based speedup technique described here, we computed Hamming distance between a pair of descriptors only when k out of five sketches were identical. Figure 15.7b shows running time for $k = 2$ and 3 and shows how the min-hash strategy provides an order of magnitude speedup over the case when exact linear scan is performed using Hamming distance on BRIEF descriptors. In all our location recognition experiments, k was set to 2.

15.4 Conclusion

We have described a new discriminative approach for recognizing urban landmarks and locations that exploits approximate GPS location to train compact classifiers. The classifiers can also be efficiently evaluated at query time. The accuracy of our method is competitive with that of state of the art methods despite the compact representation. The small download footprint of the discriminative dictionary and the classifier as well as the efficient dense matching strategy makes the approach feasible for a location recognition system designed to run solely on a mobile device.

References

1. Aly M, Welinder P, Munich M, Perona P (2009) Towards automated large scale discovery of image families. CVPR Workshop Intern Vis 9–16
2. Amit YDG (1997) Shape quantization and recognition with randomized trees. Neural Comput 9
3. Arth C, Schmalstieg D (2011) Challenges of large-scale augmented reality on smartphones. Graz University of Technology, Graz, pp 1–4
4. Arth C, Wagner D, Klopschitz M, Irschara A, Schmalstieg D (2009) Wide area localization on mobile phones. In: ISMAR, pp 73–82
5. Arth C, Klopschitz M, Reitmayr G, Schmalstieg D (2011) Real-time self-localization from panoramic images on mobile devices. In: 2013 IEEE international symposium on mixed and augmented reality (ISMAR) vol 0, pp 37–46
6. Baatz G, Koser K, Grzeszczuk R, Pollefeys M (2010) Handling urban location recognition as a 2d homothetic problem. In: IEEE proceedings of ECCV
7. Baatz G, Saurer O, Köser K, Pollefeys M (2012) Large scale visual geo-localization of images in mountainous terrain. In: ECCV (2), pp 517–530
8. Bergamo A, Sinha SN, Torresani L (2013) Leveraging structure from motion to learn discriminative codebooks for scalable landmark classification. In: CVPR, pp 763–770
9. Breiman L (2001) Random forests. Machine Learn 45
10. Cstrecha AM, Bronstein MMB, Fua P (2012) LDAHash: improved matching with smaller descriptors. IEEE Trans Pattern Anal Mach Intell 34(1)
11. Calonder M, Lepetit V, Strecha C, Fua P (2010) BRIEF: Binary robust independent elementary features. In: ECCV 4:778–792
12. Cao S, Snavely N (2013) Graph-based discriminative learning for location recognition. In: CVPR, pp 700–707
13. Cao S, Snavely N (2014) Minimal scene descriptions from structure from motion models. In: CVPR
14. Chandrasekhar V, Takacs G, Chen D, Tsai S, Grzeszczuk R, Girod B (2009) CHoG: compressed histogram of gradients a low bit-rate feature descriptor. In: IEEE conference on computer vision and pattern recognition (2009), pp 2504–2511
15. Chen DM, Baatz G, Koser K, Tsai SS, Vedantham R, Pylvanainen T, Roimela K, Chen X, Bach J, Pollefeys M, Girod B, Grzeszczuk R (2011) City-scale landmark identification on mobile devices. In: 2013 IEEE conference on computer vision and pattern recognition, vol 0, pp 737–744
16. Cheng Z, Ren J, Shen J, Miao H (2013) Building a large scale test collection for effective benchmarking of mobile landmark search. In: Advances in multimedia modeling, pp 36–46. Springer
17. Crandall D, Owens A, Snavely N, Huttenlocher D (2011) Discrete-continuous optimization for large-scale structure from motion. In: CVPR, pp 3001–3008
18. Criminisi A, Shotton J, Konukoglu E (2012) Decision forests: a unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Found Trends Comput Graph Vis 7(2–3):81–227
19. Doersch C, Singh S, Gupta A, Sivic J, Efros AA (2012) What makes paris look like paris? ACM Trans Graph 31(4)
20. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24:381–395
21. Gronat P, Obozinski G, Sivic J, Pajdla T (2013) Learning and calibrating per-location classifiers for visual place recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition
22. Guan T, Fan Y, Duan L, Yu J (2014) On-device mobile visual location recognition by using panoramic images and compressed sensing based visual descriptors. PloS one 9(6):e98,806
23. Hays J, Efros A (20078) IM2GPS: estimating geographic information from a single image. In: IEEE proceedings of CVPR

24. http://maps.google.com/help/maps/streetview/
25. http://www.bing.com/maps/
26. Hua G, Brown M, Winder S (2007) Discriminant embedding for local image descriptors. In: IEEE proceedings of ICCV
27. Irschara A, Zach C, Frahm JM, Bischof H (2009) From structure-from-motion point clouds to fast location recognition. In: CVPR, pp 2599–2606. IEEE
28. Jshotton M, Johnson RC (2008) Semantic texton forests for image categorization and segmentation. In: IEEE proceedings of CVPR
29. Knopp J, Sivic J, Pajdla T (2010) Avoiding confusing features in place recognition. In: IEEE proceedings of ECCV
30. Lepetit V, Fua P (2006) Keypoint recognition using randomized trees. PAMI 28:1465–1479
31. Li X, Wu C, Zach C, Lazebnik S, Frahm JM (2008) Modeling and recognition of landmark image collections using iconic scene graphs. In: IEEE proceedings of ECCV
32. Li Y, Crandall D, Huttenlocher D (2009) Landmark classification in large-scale image collections. In: IEEE Proceedings of ICCV
33. Li Y, Snavely N, Huttenlocher D (2010) Location recognition using prioritized feature matching. In: IEEE Proceedings of ECCV
34. Li Y, Snavely N, Huttenlocher D, Fua P (2012) Worldwide pose estimation using 3d point clouds. In: Computer Vision–ECCV 2012, pp 15–29. Springer
35. Li Z, Yap KH (2012) Content and context boosting for mobile landmark recognition. IEEE Sig Process Lett 19(8):459–462
36. Lim H, Sinha SN, Cohen MF, Uyttendaele M (2012) Real-time image-based 6-dof localization in large-scale environments. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR), pp 1043–1050. IEEE
37. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60
38. Micusík B, Wildenauer H, Kosecka J (2008) Detection and matching of rectilinear structures. In: IEEE Proceedings of CVPR
39. Middelberg S, Sattler T, Untzelmann O, Kobbelt L (2014) Scalable 6-dof localization on mobile devices. In: Computer vision ECCV 2014, lecture notes in computer science, vol 8690, pp 268–283
40. Nister D, Stewenius H (2006) Scalable recognition with a vocabulary tree. In: CVPR, pp 2161–2168
41. Ondrej Chum JP, Zisserman A (2008) Near duplicate image detection: min-hash and tf-idf weighting. In: BMVC
42. Perdoch OCM, Matas J (2009) Geometric min-hashing: Finding a (thick) needle in a haystack. In: IEEE Proceedings of CVPR
43. Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2007) Object retrieval with large vocabularies and fast spatial matching. In: IEEE proceedings of CVPR
44. Robertson D, Cipolla R (2004) An image based system for urban navigation. In: BMVC, pp 819–828
45. Sattler T, Leibe B, Kobbelt L (2012) Improving image-based localization by active correspondence search. In: ECCV 2012, pp 752–765. Springer
46. Schindler G, Brown M, Szeliski R (2007) City-scale location recognition. In: IEEE proceedings of CVPR
47. Shao H, Svoboda T, Gool LV (2003) ZUBUD-Zurich buildings database for image based recognition. Tech. rep., No. 260, Swiss Federal Inst. of Technology
48. Tola E, Lepetit V, Fua P (2010) DAISY: an efficient dense descriptor applied to wide baseline stereo. IEEE transactions on pattern analysis and machine intelligence 32(5):815–830
49. Torii A, Sivic J, Pajdla T, Okutomi M (2013) Visual place recognition with repetitive structures. In: Proceedings of the IEEE conference on computer vision and pattern recognition
50. Turcot P, Lowe DG (2009) Better matching with fewer features: the selection of useful features in large database recognition problems. In: ICCV workshop on emergent issues in large amounts of visual data (WS-LAVD)

51. Weston J, Mukherjee S, Chapelle O, Pontil M, Poggio T, Vapnik V (2000) Feature selection for SVMs. In: Advances in neural information processing systems, vol 13, pp 668–674. MIT Press
52. Winder SAJ, Hua G, Brown M (2009) Picking the best daisy. In: CVPR, pp 178–185
53. Zamir A, Shah M (2010) Accurate image localization based on google maps street view. In: IEEE proceedings of ECCV
54. Zhang W, Kosecka J (2007) Hierarchical building recognition. Image Vis Comput 25(5):704–716