# Graph-Based Discriminative Learning for Location Recognition

Song Cao Noah Snavely
Cornell University

## Abstract

*Recognizing the location of a query image by matching it to a database is an important problem in computer vision, and one for which the* representation *of the database is a key issue. We explore new ways for exploiting the structure of a database by representing it as a graph, and show how the rich information embedded in a graph can improve a bag-of-words-based location recognition method. In particular, starting from a graph on a set of images based on visual connectivity, we propose a method for selecting a set of subgraphs and learning a local distance function for each using discriminative techniques. For a query image, each database image is ranked according to these local distance functions in order to place the image in the right part of the graph. In addition, we propose a probabilistic method for increasing the diversity of these ranked database images, again based on the structure of the image graph. We demonstrate that our methods improve performance over standard bag-of-words methods on several existing location recognition datasets.*

## 1. Introduction

Location recognition—determining where an image was taken—is an important problem in computer vision. However, there is no single definition for what it means to be a location, and, accordingly, a wide variety of representations for places have been used in research: Are places, for instance, a set of distinct landmarks, each represented by a set of images? [33, 16] Are places latitude and longitude coordinates, represented with a set of geotagged images? [11] Should places be represented with 3D geometry, from which we can estimate an explicit camera pose for a query image? [17, 25, 18] This question of *representation* has analogues in more general object recognition problems, where many approaches regard objects as belonging to pre-defined categories (cars, planes, bottles, etc.), but other work represents objects more implicitly as structural relations between images, encoded as a graph (as in the Visual Memex [19]).

Inspired by this latter work, our paper addresses the location recognition problem by representing places as graphs encoding relations between images, and explores how this representation can aid in recognition. In our case, graphs represent visual overlap between images—nodes correspond to images, and edges to overlapping, geometrically consistent image pairs—leveraging recent work on automatically building image graphs (and 3D models) from large-scale image collections [1, 7, 4, 3]. An example image graph for photos of the town of Dubrovnik is shown in Figure 1. Given an image graph, our goal is to take a query image and plug it in to the graph in the right place, in effect recognizing its location. The idea is that the structure inherent in these graphs encodes much richer information than the set of database images alone, and that utilizing this structural information can result in better recognition methods.

We make use of this structural information in a bag-of-words-based location recognition framework, in which we take a query image, retrieve similar images in the database, and perform detailed matching to verify each retrieved image

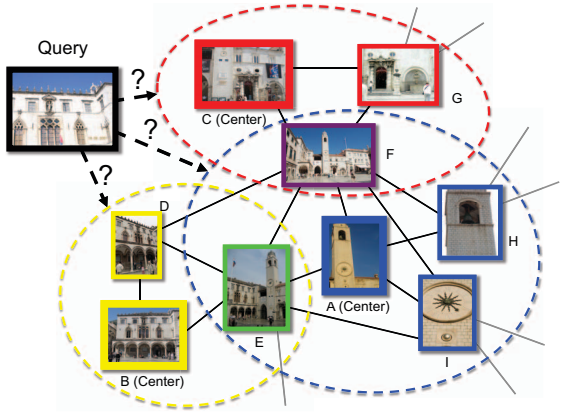


Figure 1. **A segment of an example image matching graph with three clusters defined by representative images A, B and C.** Nodes in this graph are images, and edge connect overlapping images. In order to match a new query image to the graph, our method learns local distance functions for a set of neighborhoods that cover the graph, for instance, the neighborhoods centered at nodes A, B, and C, circled with colored boundaries. Given a query image, we match to the graph using these learned neighborhood models, rather than considering database images individually. Each neighborhood has its own distinctive features, and our goal is to learn and use them to aid recognition.

until a match is found. While others have used image graphs in various settings before (especially in 3D reconstruction), our main contribution is to introduce two new ways to exploit the graph's structure in recognition. First, we **build local models** of what it means to be similar to each neighborhood of the graph (Figure 1). To do so, we use the graph's structure to define sets of images that are similar, and sets that are different, and use discriminative learning techniques to compute local distance functions tuned to specific parts of the graph. Second, we use the connectivity of the graph to **encourage diversity** in the set of results, using a probabilistic algorithm to retrieve a shortlist of similar images that are more likely to have at least one match. We show that our graph-based approach results in improvements over bag-of-words retrieval methods, and yields performance that is close to more expensive direct feature matching techniques on existing location recognition datasets.

## 2. Related Work

**Image retrieval and supervised learning.** As with other location recognition approaches [27, 12, 14, 26], our work uses an image-retrieval-based framework using a bag-of-words model for a database of images. However, our goal is not retrieval per se (i.e., to retrieve all related instances of a query image), but instead recognition, where we aim to determine where an image was taken (for which a single correctly retrieved database image can be sufficient).

Our work uses supervised learning to improve on such methods. Prior work has also used various forms of supervision to improve bag-of-words-style methods for both retrieval and recognition. One type of supervision is based on *geolocation*; images that are physically close—on the same street, say—should also be closer in terms of their image distance than images across the city or the globe. Geolocation cues have been used to reweight different visual words based on their geographic frequency [27, 14], or to find patches that discriminate different cities [6]. Other methods rely on image matching to identify good features, as we do. Turcot and Lowe [31] perform feature matching on database images to find reliable features. Arandjelovic and Zisserman propose *discriminative query expansion* in which a per-query-image distance metric is learned based on feedback from image retrieval [2]. Mikulik *et al*. use image matches to compute global correlations between visual words [21]. In contrast, we use discriminative learning to learn a set of *local* distance metrics for the database as a pre-process (rather than at query time), leveraging the known graph structure of the database images.

**Representing places.** Places in computer vision are often represented as sets of images (e.g., the Eiffel Tower can be represented with a collection of photos [33]). However, many other representations of places have been explored. Some methods use *iconic images* to represent sets of images taken from very similar viewpoints [15, 13]. Other approaches use 3D point clouds, derived from structure from motion, as a richer geometric representation of a place [17, 24]. Closer to our approach are methods that explicitly construct and exploit image graphs. For instance, Torii *et al*. download Google Streetview images to form a recognition database, and leverage the underlying Street View image network; in their approach, they take linear combinations of neighboring images (in bag-of-words space) to more accurately recognize the continuum of possible viewpoints [30]. Li *et al*. use a visibility graph connecting images and 3D points in a structure-from-motion model to reason about point co-occurrence for location recognition [18]. A main contribution of our approach is to combine the power of discriminative learning methods with the rich structural information in an image graph, in order to learn a better database representation and to better analyze results at query time.

## 3. Graph-based Location Recognition

We base our algorithm on a standard bag-of-words framework [29], with images represented as $L_2$ normalized histograms of visual words, using a large vocabulary trained from SIFT descriptors. Our problem takes as input a database of images $\mathcal{I}$ represented as bag-of-words vectors, and an image graph $G$, with a node for each image $a \in \mathcal{I}$, and edges $(a, b)$ connecting overlapping, geometrically consistent image pairs. Our goal is to take a new query image and predict which part of the graph this image is connected to, then use this information to recognize its location.

To achieve this goal, we use the query to retrieve a shortlist of similar database images, and perform detailed matching and geometric verification on the top few matches. Because our goal is recognition, rather than retrieval, we want to have at least one correct match appear as close as possible to the top of the shortlist (rather than retrieve all similar images). Towards that end, our method improves on the often noisy raw bag-of-words similarity measure by leveraging the graph in two ways: (1) we discriminatively learn local distance functions on neighborhoods of the image graph (Section 3.2), and (2) we use the graph to generate a ranked list that encourages more diverse results (Section 3.3).

### 3.1. Image Matching Graphs

We construct an image graph for the database using a standard image matching pipeline [1]: we extract features from each image, and, for a set of image pairs, find nearest neighbor features and perform RANSAC-based geometric verification. These matches are sufficient for our method (though to improve the quality of the matching, we can also run structure from motion to obtain a point cloud and a refined set of image correspondences). For each image pair $(a, b)$ with sufficient inliers matches, we create an edge in

our graph $G$. We also save the number of inliers $N(a,b)$ for each image pair to derive edge weights for the graph. In our experience, the graphs we compute have very few false edges—almost all of the matching pairs are correct—though there may be edges missing from the graph because we do not exhaustively test all possible edges.

In parts of our algorithm, we will threshold edges by their weights, discarding all edges below a threshold. The edge weights we define are related to the idea of a *Jaccard index*; we define a weight $J(a,b) = \frac{N(a,b)}{N(a)+N(b)-N(a,b)}$, where $N(a)$ and $N(b)$ denote the total number of points seen in $a$ and $b$ respectively. This measures the similarity of the two images as the number of features $N(a,b)$ they have in common, normalized by the union of their feature sets. This measure ranges from 0 to 1; 0 if no overlap, and 1 if every feature was matched. This normalization reduces bias towards images with large numbers of features.

### 3.2. Graph-based Discriminative Learning

How can we use the information encoded in the graph to better recognize the location of a query image? We first address this problem as one of distance (or similarity) metric learning. There are many possible ways to learn a metric for the images in the graph. For example, one could take all the connected pairs in the graph to be positive examples and the other pairs as negative examples, to learn a single, global distance metric for a specific dataset [3]. At the other extreme, one could learn a distance metric for each image in the database, analogous to how Exemplar SVMs have been used for object detection [20].

We tried both approaches, but found that we achieved better performance with approach somewhere in the middle of these two extremes. In particular, we divide the graph into a set of overlapping subgraphs, and learn a separate distance metric for each of these representative subgraphs. Our approach, then, consists of the following steps:

**At Training Time**

**1.** Compute a covering of the graph with a set of subgraphs.

**2.** Learn and calibrate an SVM-based distance metric for each subgraph.

**Ay Query Time**

**3.** Use the models in Step 2 to compute the distance from a query image to each database image, and generate a ranked shortlist of possible image matches.

**4.** Perform geometric verification with the top database images in the shortlist.

We now describe each step in more detail. Later, in Section 3.3, we discuss how we improve Step 3 by reranking the shortlist based on the structure of the graph.

**Step 1: Selecting representative neighborhoods.** We start by covering the graph with a set of representative subgraphs; afterwards, for each subgraph, we will learn a local similarity function, using the images in the subgraph as positive examples, and other, unrelated images in the graph as negative examples. What makes a good subgraph? We want each subgraph to contain images that are largely similar, so that our learning problem has a relatively compact set of positive example images that can be explained with a simple model. On the other hand, we also want as many positive examples as possible, so that our models have enough data from which to generalize. Finally, we want our subgraphs to completely cover the graph (i.e., each node is in at least one subgraph), so that we can build models that apply to any image of the location modeled in the database.

Based on these criteria, we cover the graph by selecting a set of representative *exemplar* images, and defining their (immediate) neighborhoods as subgraphs in a graph cover, as illustrated in Figure 1. Formulated this way, the covering problem becomes one of selecting a set of representative images that form a *dominating set* of the graph. For a graph $G$, and a set of exemplar images $C$, we say an image $a \in \mathcal{I}$ is covered by $C$ if either $a \in C$, or $a$ is adjacent to an image in $C$. If $C$ covers all nodes, then $C$ is a dominating set. We would like $C$ to be as small as possible, and accordingly, the neighborhood of each node in $C$ to be as large as possible. Hence, we seek a *minimum* dominating set. Such sets have been used before for 3D reconstruction [10]; here we use them to define a set of classifiers.

Finding an exact minimum dominating set is an NP-complete problem. We use a simple greedy algorithm to find an approximate solution [9]. Starting with an empty set, we iteratively choose the image that covers the maximum number of as-yet uncovered images in the graph, until all images are covered. Figure 2 shows an example image graph for the Dubrovnik dataset [17] and the exemplar images selected by our algorithm.

**Step 2a: Discriminative learning on neighborhoods.** For each neighborhood selected in Step 1, the next step is to learn a classifier that will take a new image, and classify it as belonging to that neighborhood or not. We learn these classifiers using standard linear SVMs on bag-of-words histograms, one for each neighborhood, and calibrate the set of SVMs as described in Step 2b; at query time, these classifiers will be used to define a set of similarity functions for ranking the database images given a query image. This use of classifiers for ranking has found many applications in vision and machine learning, for instance in image retrieval using local distance functions [8] or Exemplar SVMs [28].

First, for each neighborhood around an exemplar node $c \in C$, we must define a set of positive and negative example images as training data for the SVM. To define the positive set, we simply use the images in the neighborhood. For this task, we found that thresholding the edges in the graph by their weight—applying a stricter definition of connec-

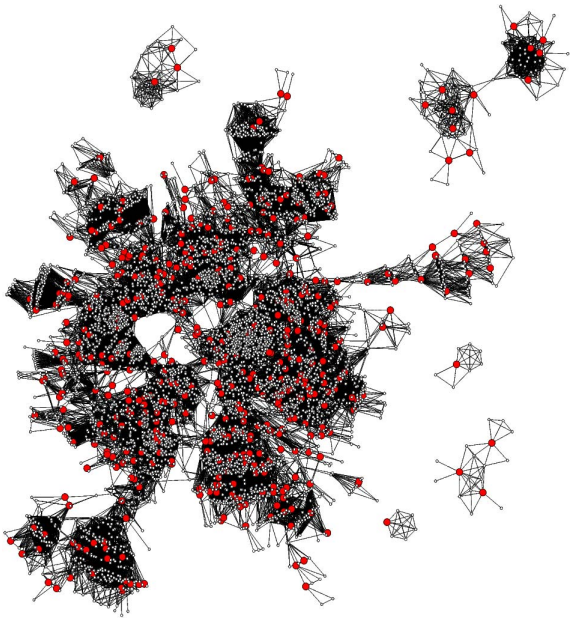

Figure 2. **Image matching graph for the Dubrovnik dataset.** This graph contains 6,844 images; the large, red nodes denote representative images selected by our covering algorithm (478 images in total). Although the set of representative images is much smaller than the entire collection, their neighborhoods cover the matching graph. For each neighborhood, we learn a classifier for determining whether a new image belongs to that neighborhood.

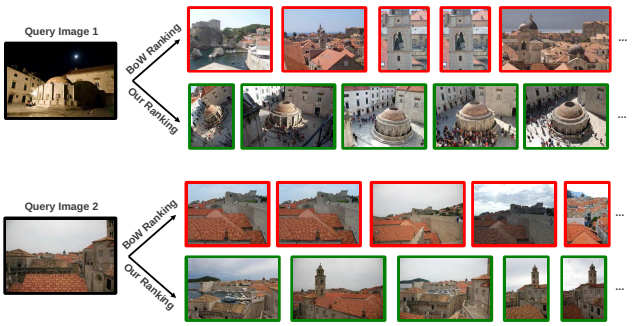


Figure 3. **Two example query images and their top 5 ranked results of our method and raw tf-idf retrieval.** For each result, a green border indicates a correct match, and a red border indicates an incorrect match. These two example query images are difficult for BoW retrieval techniques, due to drastically different lighting conditions (query image 1) and confusing features (rooftops in query image 2). However, with our discriminatively learned similarity functions, correctly matching images are ranked higher than with the baseline method.

tivity, and yielding more compact neighborhoods—yielded better classifiers than using all edges found by the image matching procedure. To define the negative set for the neighborhood around an exemplar $c$, we first find a small set of *hard negatives*—images with high BoW similarities to $c$, but not in its neighborhood. These hard negatives are combined with other randomly sampled non-neighboring images in the graph to form a negative set. Here we use the original, as opposed to thresholded, graph to define connectivity, to minimize the chances of including a false negative in the negative set. In this way, the image graph $G$ gives us the supervision necessary to define positives and negatives for learning, just as geotags have provided a supervisory cue for discriminative location recognition in previous work [27, 14].

Given the training data for each neighborhood, we learn a linear SVM to separate neighborhood images from non-neighborhood images, using the tf-idf weighted, $L_2$ normalized bag-of-words histograms for each image as features. We randomly split the training data into training and validation subsets for parameter selection in training the SVM (more details in Section 4.2). For each neighborhood centered on exemplar $c$, the result of training is an SVM weight vector $\mathbf{w}_c$ and a bias term $b_c$. Given a new query image, represented as a bag-of-words vector $\mathbf{q}$, we can compute the decision value $\mathbf{w}_c \cdot \mathbf{q} + b_c$ for each exemplar image $c$.

**Step 2b: Calibrating classifier outputs.** Since our classifiers are independently trained, we need to normalize their outputs before comparing them. To do so, we convert the decision value of each SVM classifier into a probability value, using Platt's method [23] on the whole set of training data. For a neighborhood around exemplar $c$, and a query image vector $\mathbf{q}$, we refer to this probability value as $P_c(\mathbf{q})$.

**Step 3: Generating a ranked list of database images.** For a query image represented as a BoW vector $\mathbf{q}$, we can now compute a probability of $\mathbf{q}$ belonging to the neighborhood of each exemplar image $c$. Using these values, it is straightforward to generate a ranked list of the exemplar images $c \in C$ by sorting by $P_c(\mathbf{q})$ in decreasing order. However, we found that just verifying the query image against exemplar images sometimes failed simply because the exemplar images represent a much sparser set of viewpoints than the full graph. Hence, we would like to create a ranked list of *all* database images. To do so, we take the sorted set of neighborhoods given by the probability values, and then we sort the images *within* each neighborhood by their original tf-idf similarity. We then concatenate these per-neighborhood sorted lists; since a database image can appear in multiple overlapping neighborhoods (see Figure 1), in the final list it appears only in list of the best-ranked neighborhood. This results in a ranking of the entire list of database images.

**Step 4: Geometric verification.** Finally, using the ranking of database images from Step 3, we perform feature matching and RANSAC-based geometric verification between the query image and each of the images in the shortlist in turn, until we find a true match. If we have a 3D structure from motion model, we can then associate 3D points with matches

in the query image, and determine its pose [18]. If not, we can associate the location of the matching database image as the approximate location of the query image. Because feature matching and verification is relatively computationally intensive, the quality of the ranking from Step 3 highly impacts the efficiency of the system—ideally, a correct match will be among the top few matches, if not the first match.

Using this simple approach, we observe improvements in our ranked lists over raw BoW retrieval results, as shown in the examples in Figure 3. In particular, the top image in the ranked list is more often correct. However, when the top ranked cluster is incorrect, this method has the effect of saturating the top shortlist with similar images that are all wrong—there is a lack of *diversity* in the list, with the second-best cluster pushed further down the list. To avoid this, we propose several methods to encourage a diverse shortlist of images.

### 3.3. Improving the Shortlist

In this section, we first introduce a probabilistic method that uses the graph to introduce more diversity into the shortlist, increasing the likelihood of finding a correct match among the top few retrieved images. Second, we demonstrate several techniques to introduce regularization using BoW ranking to further improve recognition performance.

**Probablistic Reranking.** Our problem is akin to the well-known Web search ranking problem (as opposed to standard image retrieval). Rather than retrieve *all* instances relevant to a given query, we want to retrieve a small set of results that are both *relevant* and *diverse* (see Figure 4 for an example), so as to cover multiple possible hypotheses—just as a Web search for the term "Michael Jordan" might productively return results for both the basketball player and the machine learning researcher. While introducing diversity in Web search has been studied in the machine learning literature [32], we are unaware of it being used in location recognition; in our problem, it is the automatic verification procedure that is examining results, rather than a human. To introduce diversity, we propose a probabilistic approach for reranking the shortlist. The idea is, in some ways, the converse of *query expansion* on positive matches to increase recall in image retrieval. In our case, we use *negative evidence* to increase the pool of diverse matches. For instance, in the case where the first retrieved image is *not* a match to the query, we want to select the second image conditioned on this outcome, perhaps selecting an image dissimilar to this first match (and similarly for the third image conditioned on the first two being incorrect). How can we compute such conditional probabilities? We again turn to the image graph.

First, some terminology. For a database image $a$, we define a random variable $X_a$ representing the event that the query image matches image $a$; $X_a = 1$ if image $a$ is a match, and 0 otherwise. Thus, using the notation above, $P_c = P(X_c = 1)$ for an exemplar image $c$, and similarly $P_a = P(X_a = 1)$ for *any* database image, using the simple heuristic above that a non-exemplar database image takes the maximum probability of all neighborhoods it belongs to. As before, we choose the database image $a$ with the highest $P_a$ as the top-ranked image. However, to select the *second* ranked image, we are instead more interested in the conditional probability $P'_b = P(X_b = 1|X_a = 0)$ than its raw appearance-based probability $P(X_b = 1)$ alone. We can compute this conditional probability as:

$$\begin{aligned} P'_b = P(X_b = 1|X_a = 0) &= \frac{P(X_b = 1, X_a = 0)}{P(X_a = 0)} \\ &= \frac{P(X_b = 1) - P(X_b = 1, X_a = 1)}{1 - P(X_a = 1)} \\ &= \frac{P_b - P(X_b = 1|X_a = 1)P(X_a = 1)}{1 - P_a} \\ &= \frac{P_b - P_{ba}P_a}{1 - P_a} = P_b \left( \frac{1 - \frac{P_{ba}}{P_b} P_a}{1 - P_a} \right) \end{aligned} \quad (1)$$

where $P_{ba} = P(X_b = 1|X_a = 1)$ denotes the conditional probability that image $b$ matches the query given that image $a$ matches. The last line in the derivation above relates $P'_b$ to $P_b$ via an *update factor*, $(1 - \frac{P_{ba}}{P_b} P_a)/(1 - P_a)$, that depends on $P_a$ (the probability that the top ranked image matches) and $P_{ba}$ (a conditional probability). We use the image graph to estimate $P_{ba}$, the intuition being that the more similar $b$ is to $a$—i.e., stronger the connection between $a$ and $b$ in the graph—the higher $P_{ba}$ should be. In particular, we estimate $P_{ba}$ as $\frac{N(a,b)}{N(a)}$, the ratio of the number of shared features between $a$ and $b$ divided by the total number of feature points in $a$. Note that in general $P_{ab} \not\equiv P_{ba}$, i.e., this similarity measure is asymmetric. These measures are precomputed, along with the Jaccard indices $J(a, b)$ described in Section 3.1.

The update factor in Eq. (1) has an intuitive interpretation: if image $b$ is very similar to image $a$ according to the graph (i.e., $P_{ba}$ is large), then its probability score is downweighted (because if $a$ is an incorrect match, then $b$ is also likely incorrect). On the other hand, if $b$ is not connected to $a$, its score will tend to be boosted. However, we do not want to apply this update too quickly, for fear of downweighting many images based on the evidence of a single mismatch. To regulate this factor, we introduce a parameter $\alpha$, and define a regularized update factor $(1 - \alpha \frac{P_{ba}}{P_b} P_a)/(1 - \alpha P_a)$. If $\alpha = 0$, the update has no influence on the ranking result, and when $\alpha = 1$, it has its full effect. We use $\alpha = 0.9$ in our experiments. We iteratively choose the image $b$ with the highest updated score $P'_b$ and recalculate scores using (1).

**BoW Regularization.** Our learned discriminative models often perform well, but we observed that for some rare query images, our models consistently perform poorly (perhaps due


Figure 4. **An example query image and the top 5 ranking results using our method with and without probablistic ranking.** Green borders indicate correct matches, and red borders incorrect ones. Without probabilistic ranking, our algorithm outputs top 5 results that are similar, but incorrect. With probabilistic ranking, more diversity is encouraged in the top ranking results, leading to correct images in the top 5 results.

to sparser parts of the graph having relatively few training examples, see Figure 2). For this reason, we found it helpful to use the original tf-idf-based similarities as a way of "regularizing" our rankings, in case of query images for which our models perform poorly. We do this in three ways. First, as a simple strategy, for query images where *all* models give a probability score below a minimum threshold $P_{\min}$ (0.1 in our tests), we *fall back* to tf-idf scores, as we found low probability scores unreliable for ranking. (In our experiments, this occurs in $\sim 5\%$ of queries.) Second, to regularize our probability scores in case of overfitting, we take a weighted *average* of our probability scores and a tf-idf-based probability value; this value is given by a logistic regressor fitted using matching and non-matching image pairs in the image graph. Finally, we found that our learned models and the original tf-idf scores sometimes were complementary; while our models work well for many queries, some query images still performed better under tf-idf. Thus, as a way of introducing more diversity, and an alternative for the falling back strategy, we *interleave* the results of the two rankings. The order of interleaving is determined by the maximum value of our probability outputs, which represents the confidence of our original ranking. If this value is less than a threshold (we use 0.1), then BoW ranking goes first, and vice versa. In our experiments, we use the simple fall back strategy by default, and separately evaluate a combination of averaging and interleaving as a stronger form of tf-idf regularization.

# 4. Experiments

As discussed in Section 3.2, a key bottleneck of image retrieval-based location recognition systems is the quality of the image ranking—we want the first true match to a query image to rank as high in the list as possible, so we have to run the verification procedure on as few images as possible. Hence, we evaluate the accuracies at top $k$ ($k \in \{1, 2, 5, 10\}$), i.e., the percentage of query images that have at least one correct match in the top $k$ results. Note that all the methods we test are compared purely based on the shortlist they generate on an equal footing, without using RANSAC-based verification before examining results. We apply detailed verification in all cases by checking each short-listed image sequentially until the first true match is found, at which point a localization is achieved.

Table 1. **Summary of datasets and their neighborhoods used in our experiments.** The representative neighborhoods (clusters) are found using graphs whose edge weights are defined using *Jaccard index* and thresholded by value 0.01. The rightmost column shows the average cluster size in each dataset.

| Dataset | # Queries | # DB Imgs | # Clusters | Ave. Cluster Size |
|---|---|---|---|---|
| Dubrovnik [17] | 800 | 6,044 | 188 | 206.7 |
| Rome [17] | 1,000 | 15,179 | 352 | 293.0 |
| Aachen [26] | 369 | 4,479 | 161 | 82.0 |

## 4.1. Datasets and Preprocessing

We evaluate our algorithm on the Dubrovnik and Rome datasets [17] and the Aachen dataset [26]; these are summarized in Table 1, along with statistics over the neighborhoods we compute. To represent images as BoW histograms, we learn two kinds of visual vocabularies [22]: one vocabulary learned from each dataset itself (a **specific** vocabulary) and another shared vocabulary learned from ∼20,000 randomly sampled images from an unrelated dataset (a **generic** vocabulary). Each vocabulary is of size 1M. As our ground truth, we count an image pair as matching if they have at least 12 inlier matches.

## 4.2. Performance Evaluation

**Baselines.** For all datasets, we compare (a) standard tf-idf image retrieval [22] and (b) its probabilistic reranked version, (c) our learning-based technique, and (d) our learning method using diversity reranking as well as (e) strong BoW regularization. We note that our method is orthogonal to many other improvements to bag-of-words models [2], as we can generalize to more sophisticated feature representations. In addition, for one dataset (Dubrovnik, with a specific vocabulary), we also compare to a range of other baselines, including a more recent retrieval method using co-occuring sets of visual words [5] and a GPS-based baseline inspired by [14, 27]. For the latter, we randomly select a set of exemplar images, define the nearest neighbors using GPS positions as positives and the rest as negatives and use the same learning and retrieval techniques described above thereafter. Finally, we evaluate two alternative learning approaches: a global distance metric learned using pairs of matching and non-matching image pairs in the graph [3], and our technique but trained using *every* database image as a center (i.e., learning a per-image distance metric).

**Experiment details.** From a Jaccard index weighted image graph (thresholded by 0.01), we choose exemplar images

Table 2. **Recognition accuracies on all datasets.** "GBP" stands for our graph-based probability ranking; "+RR" stands for our probabilistic reranking; "+BoW" stands for strong regularization using BoW ranking. Here we show results using both specific and generic vocabularies for Dubrovnik dataset, and only using specific vocabulary for others, whose generic cases show similar trend (slightly worse) compared to specific cases.

Dubrovnik (Specific Vocab.)

| Method | top1 | top2 | top5 | top10 | mAP |
|---|---|---|---|---|---|
| BoW [29] | 87.50% | 92.75% | 97.62% | 98.50% | 0.401 |
| BoW+RR | 87.50% | 93.38% | 96.63% | 97.50% | 0.058 |
| Co-ocset [5] | 87.50% | 92.50% | 97.50% | 98.62% | 0.389 |
| GPS Model | 87.87% | 89.75% | 91.75% | 93.25% | 0.367 |
| Global Model [3] | 85.37% | 91.63% | 95.87% | 97.38% | **0.643** |
| Instance Model | 90.00% | 95.13% | 98.12% | 98.50% | 0.643 |
| GBP | **94.38%** | 96.37% | 98.25% | 98.50% | 0.626 |
| GBP+RR | **94.38%** | 96.25% | 98.62% | 99.13% | 0.273 |
| GBP+RR+BoW | 94.25% | **97.12%** | **99.37%** | **99.50%** | 0.122 |

Dubrovnik (Generic Vocab.)

| Method | top1 | top2 | top5 | top10 | mAP |
|---|---|---|---|---|---|
| BoW | 75.88% | 83.00% | 90.88% | 95.63% | 0.512 |
| BoW+RR | 75.88% | 83.62% | 93.25% | **96.25%** | 0.065 |
| GBP | 81.25% | 85.13% | 88.13% | 90.00% | **0.512** |
| GBP+RR | 81.25% | 83.87% | 89.88% | 95.13% | 0.151 |
| GBP+RR+BoW | **81.88%** | **90.00%** | **94.00%** | 96.00% | 0.085 |

Rome

| Method | top1 | top2 | top5 | top10 | mAP |
|---|---|---|---|---|---|
| BoW | 97.40% | 98.50% | 99.50% | 99.60% | 0.674 |
| BoW+RR | 97.40% | 98.70% | 99.10% | 99.10% | 0.047 |
| GBP | 97.80% | 98.70% | 99.30% | 99.30% | **0.789** |
| GBP+RR | 97.80% | **98.80%** | 99.30% | 99.70% | 0.403 |
| GBP+RR+BoW | **97.90%** | **99.00%** | **99.70%** | **99.70%** | 0.259 |

Aachen

| Method | top1 | top2 | top5 | top10 | mAP |
|---|---|---|---|---|---|
| BoW | 80.76% | 83.47% | 86.45% | 88.35% | 0.431 |
| BoW+RR | 80.76% | 82.66% | 86.45% | 88.62% | 0.069 |
| GBP | **82.38%** | 84.55% | 86.72% | 88.35% | **0.459** |
| GBP+RR | **82.38%** | 83.74% | 87.26% | 88.89% | 0.205 |
| GBP+RR+BoW | **82.38%** | **84.82%** | **88.08%** | **89.16%** | 0.185 |

(neighborhoods) and learning SVMs and logistic functions as described in Section 3. For each cluster, we use all the available positive examples (i.e. cluster sizes in Table 1), and sample roughly 5 times more negative examples. $\frac{1}{3}$ of total training data is held out for validation, and all training data is used for logistic regressor training. For each query image, we compute the estimated probability of it matching all clusters, and obtain the initial ranking of the database images as described in Section 3.2. We show the results of our method (a) ranking with just the probability scores, (b) reranking using our diversity measure, and (c) strong BoW regularization using tf-idf scores (through both averaging the two scores, with a weight of $\frac{5}{6}$ on our score, and $\frac{1}{6}$ on the tf-idf-based probability score, and interleaving two resulting rankings as described in Section 3.3).

**Results.** The results are shown in Table 2. From Dubrovnik (Specific Vocab.), we can see that the unsupervised methods (BoW, BoW+RR and Co-ocset) perform similarly; the GPS based model (GPS Model) performs better at top1 but worse for others, probably due to less accurate choices of training examples compared to those based on image graph; the globally trained classifiers (Global Model) actually perform *worse*, in general, compared to the unsupervised methods, at least as measured by how often a correct result is in the top-$k$ matches. Interestingly, however, it does improve the mAP (mean average precision) score the most, suggesting that they are better at globally ranking the images than they are at our recognition task. The per-image classifiers ("Instance Model") perform best among the baselines, but still worse than our method. We believe this is due to the nature of image graphs for unstructured collections, where some nodes have many neighbors, and others (e.g. very zoomed-in images) have only a few; training and calibration for these low-degree nodes may result in models that overfit the data and contaminate the global ranking. In addition, increasing the diversity (+RR) and strong regularization using BoW results (+BoW) both are beneficial in improving our original ranking results (though these techniques result in a smaller mAP score; this again suggests an interesting tradeoff between retrieval and recognition performance).

Similar trends follow for other datasets as well. The generic vocabulary performs worse than the specific one in general. Our cluster-based probability scores (GBP) alone consistently improve results for the top1 and top2 rankings (anywhere from a negligible amount for the Rome dataset, to $> 6\%$ for the Dubrovnik dataset with a specific vocabulary for the top1). However, the performance of GBP results increases much more slowly than the baseline tf-idf ranking as a function of $k$, and for the top10 rankings the learning approach performs worse in some cases. However, once we reintroduce diversity through probabilistic reranking (RR), our results improve slightly in general for larger rankings (1.68% on average across our datasets for top10). Additional gains are seen when regularizing our learned results with the tf-idf scores (0.38% on average for top10).

We note that the Dubrovnik dataset is more challenging than Rome, and has a more interesting graph structure (Dubrovnik spans many connected viewpoints across a city, while Rome mostly consists of distinct landmarks). Our improvement over the baselines, particularly for the top ranked image, is more apparent for Dubrovnik. For both Dubrovnik and Rome, our top 10 success rate (99.5% on Dubrovnik and 99.7% on Rome) is comparable to the results of [18] (100% / 99.7%), which uses direct 3D matching, requiring much more memory and expensive nearest neighbor computations. Our performance on Aachen dataset (89.16%) also rivals that of [26], where their best result 89.97% is achieved with a relatively expensive method, while we only use the compact set of weights learned from neighborhoods. In all cases, we improve the top $k$ accuracies over BoW retrieval techniques, resulting in a better ranking for the final step of geometric consistency check procedure.

## 5. Conclusions and Discussion

Locations are often complex and difficult to model using discrete categories or classes. We argue instead for modeling locations as graphs for recognition problems, and explore using local neighborhoods of exemplar images for learning local distance metrics. This idea could also have application in other recognition problems. Compared to raw tf-idf based location recognition, we demonstrate higher performance with little extra overhead during query time. Compared to direct matching approaches, we do not require a full 3D reconstruction and a large set of descriptors to be stored in memory. Since our approach uses a bag-of-words framework, we require less memory and have good scalability.

One limitation of our approach is that we require more memory than standard tf-idf methods, since we need to learn and use discriminative models in the database (though the number of neighborhoods we select is often an order of magnitude smaller than that of the original images (Table 1)). Another limitation is that care must be taken when training and calibrating the neighborhood models. In general, the clusters we create have relatively large variation in size, which could lead to some variation in reliability in their performance. Finding a way to automatically adjust learning parameters or synthesize the results from different clusters is an important issue, and an interesting direction of future work.

**Acknowledgements.** This work was supported in part by the NSF (grants IIS-0713185 and IIS-1111534) and Intel Corporation. We also thank Flickr users for use of their photos.

## References

[1] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building Rome in a day. In *ICCV*, 2009.

[2] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[3] S. Cao and N. Snavely. Learning to match images in large-scale collections. In *ECCV Workshop on Web-scale Vision and Social Media*, 2012.

[4] O. Chum and J. Matas. Large-scale discovery of spatially related images. *PAMI*, 2010.

[5] O. Chum and J. Matas. Unsupervised discovery of co-occurrence in sparse high dimensional data. In *CVPR*, 2010.

[6] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. A. Efros. What makes Paris look like Paris? *SIGGRAPH*, 2012.

[7] J.-M. Frahm et al. Building Rome on a cloudless day. In *ECCV*, 2010.

[8] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007.

[9] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 1998.

[10] M. Havlena, A. Torii, and T. Pajdla. Efficient structure from motion by graph optimization. In *ECCV*, 2010.

[11] J. Hays and A. Efros. Im2gps: estimating geographic information from a single image. In *CVPR*, 2008.

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

[13] E. Johns and G. Yang. From images to scenes: Compressing an image cluster into a single scene model for place recognition. In *ICCV*, 2011.

[14] J. Knopp, J. Sivic, and T. Pajdla. Avoiding confusing features in place recognition. In *ECCV*, 2010.

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

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

[17] Y. Li, N. Snavely, and D. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010.

[18] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012.

[19] T. Malisiewicz and A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. *NIPS*, 2009.

[20] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In *ICCV*, 2011.

[21] A. Mikulik, M. Perdoch, O. Chum, and J. Matas. Learning a fine vocabulary. In *ECCV*, 2010.

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

[23] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 1999.

[24] T. Sattler, B. Leibe, and L. Kobbelt. Fast image-based localization using direct 2D-to-3D matching. In *ICCV*, 2011.

[25] T. Sattler, B. Leibe, and L. Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012.

[26] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt. Image retrieval for image-based localization revisited. In *BMVC*, 2012.

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

[28] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. *SIGGRAPH ASIA*, 2011.

[29] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[30] A. Torii, J. Sivic, and T. Pajdla. Visual localization by linear combination of image descriptors. In *ICCV Workshops*, 2011.

[31] P. Turcot and D. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *Workshop on Emergent Issues in Large Amounts of Visual Data, ICCV*, 2009.

[32] Y. Yue and C. Guestrin. Linear submodular bandits and their application to diversified retrieval. In *NIPS*, 2011.

[33] Y.-T. Zheng et al. Tour the world: building a web-scale landmark recognition engine. In *CVPR*, 2009.