**Noname manuscript No.**
(will be inserted by the editor)

# Graph-Based Discriminative Learning for Location Recognition

**Song Cao · Noah Snavely**

Received: date / Accepted: date

**Abstract** Recognizing the location of a query image by matching it to an image 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 an image database by representing it as a graph, and show how the rich information embedded in such a graph can improve bag-of-words-based location recognition methods. In particular, starting from a graph based on visual connectivity, we propose a method for selecting a set of overlapping subgraphs and learning a local distance function for each subgraph 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.

**Keywords** Location recognition · discriminative learning · image graphs

## 1 Introduction

What is a place? In this paper, we consider this question in the context of the location recognition problem—determining where an image was taken—in particular, in choosing a good *representation* for places we wish to recognize. There is no single definition for what it means to be a place, and, accordingly, a wide variety of representations for places have been used in the literature: Are places, for instance, a set of discrete landmarks, each represented by a set of images? [35,18] Are places latitude and longitude coordinates, represented with a set of geotagged images? [13] Should places be represented with 3D geometry, from which we can estimate an image's location from a continuum of possible viewpoints? [19,28,20] Is a place a set of representative visual elements? [7] 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 [21]).

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 visual recognition. In our case, our 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 [1,9,5,3]. 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.

Song Cao, Cornell University
E-mail: caosong@cs.cornell.edu · Noah Snavely, Cornell University

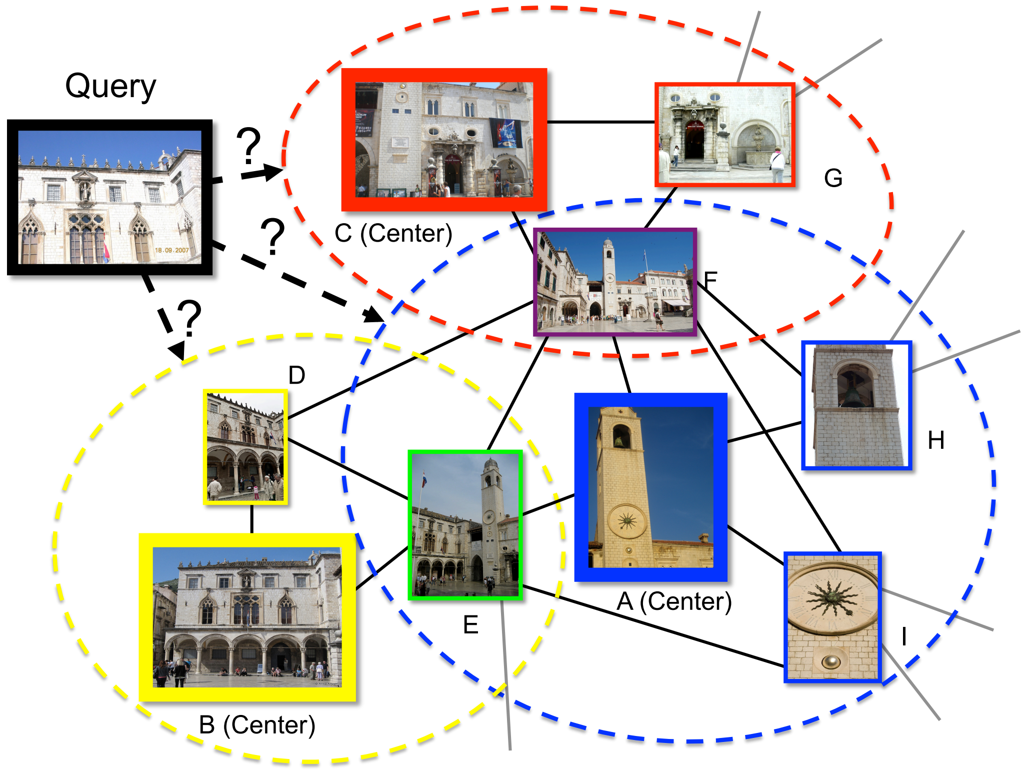

**Fig. 1 A region of an example image graph with three neighborhoods defined by representative images A, B and C.** Nodes in this graph are images, and edges connect visually overlapping images. Our method uses the graph to find a set of representative neighborhoods (inside the dotted circles above) that cover the graph, and learns a local distance function for each neighborhood. These distance functions are used to connect a new query image (left) to the rest of the graph and hence recognize its location. 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.

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 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, and hence, to specific parts of the location. Second, we use the connectivity structure 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 closer 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 [30,14,16,29], 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., we do not seek 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 according to some notion of 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 [30,16], or to find patches that are discriminative for different cities or neighborhoods [7]. Other methods rely on image matching to identify good features, as we do. Turcot and Lowe [34] 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 [23]. 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 bags of images; for instance, the Eiffel Tower can be described as a collection of photos showing that landmark [35]. However, many other representations of places have been explored. Some methods use *iconic images* to represent sets of images taken from very similar viewpoints [17,15], or otherwise select a set of representative views [14]. Other approaches use 3D point clouds, derived from structure from motion, and augmented with appearance information, as a richer geometric representation of a place [19,27]. Closest 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 make use of 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 of a city [33]. 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 [20].

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 underlying representation from the image database, and to better analyze the retrieval results at query time.

## 3 Graph-based Location Recognition

This section describes our problem setting and preliminaries, and provides an overview of our method. In our work, we take as our database a set of images $\mathcal{I}$ of a place, or set of places; the places we consider generally span a large area, such as an entire city, or multiple landmarks across a wide region. From this set of images, we first compute two types of information: **appearance** information for each image (in our work we use $L_2$-normalized bag-of-words histograms [32]), and **connectivity** information represented as an image graph $\mathcal{G}$ on the set of images $\mathcal{I}$. The graph $\mathcal{G}$ contains a node for each image $a \in \mathcal{I}$, and an edge $(a, b)$ connecting pairs of visually overlapping, geometrically consistent image pairs. Our goal is to use this structural information to help us quickly and accurately take a new query image and predict which part of the graph it is connected to, from there determine its fine-grained location within the place.

To achieve this goal, as in many image retrieval methods, we use the query to retrieve a shortlist of similar database images based on bag-of-words similarity, and perform detailed matching and geometric verification on the top few matches to (a) verify whether the match is correct and (b) determine the location of the image. Ideally, a correctly matching database image will appear near the beginning of the shortlist, if not in the top spot; because our goal is recognition, rather than retrieving all matching instances in the database, we can stop as soon as we find a correct match. Towards that end, our contribution is a method that 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 4), and (2) we use the graph to generate a ranked list that encourages more diverse results (Section 5).

First, we describe how we compute the image graph $\mathcal{G}$ from the set of images $\mathcal{I}$.

**Image Graphs.** Our method starts by constructing an image graph $\mathcal{G}$ from the image database using a standard image matching pipeline [1]. In short, we extract features from each image, and perform pairwise feature matching on a set of candidate image pairs proposed using bag-of-words image similarity. For each such pair, we find nearest neighbor features matches from one image to the other, prune these matches using Lowe's ratio test, and perform RANSAC-based geometric verification on the remaining features to compute a set of inlier matches (note that we only perform feature matching in one direction). Our method does not use these matches directly, but instead records the number of inlier feature matches for each image pair. Note that we could further improve the quality of these matches by running structure from motion to obtain a point cloud and a refined set of image correspondences, though this is not required by our method.

For each image pair $(a, b)$ with sufficient inlier matches (we use a threshold of 12 in our experiments), we create an edge in our graph $\mathcal{G}$. We also save this number of inliers, denoted $N(a, b)$, for each image pair, and use this measure to derive edge weights for the graph. In our experience, the graphs we compute have very few false positive 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.

Our algorithm also weights each edge in the graph with an estimate of the strength of the visual connection between the two images; later stages of our algorithm threshold edges by their weights. While there are many potential ways to define these weights, we found that an effective definition is one related to the idea of a *Jaccard index*. We define this weight as:

$$J(a,b) = \frac{N(a,b)}{N(a) + N(b) - N(a,b)}, \tag{1}$$

where $N(a)$ and $N(b)$ denote the number of features in $a$ and $b$ respectively that were matched to any other image (which we refer to as their *matchable* features).[1] In other words, $J(a, b)$ 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 matchable feature sets. This measure ranges from 0 to 1; 0 if no overlap, and 1 if every feature was matched between the two images. This normalization reduces bias towards images with large numbers of features. Alternatively, one could also use the raw number of features matches $N(a, b)$ as the edge weights, but we found in our experiments that the "normalized" weights defined by the Jaccard index work better in practice.

## 4 Graph-based Discriminative Learning

How can we use the information encoded in the image graph to better recognize the location of a query image? One key piece of information provided by the connectivity of the image graph is a notion of similarity—i.e., we know which images are expected to be similar (connected pairs) and which are not (disconnected pairs), according to some desired distance metric. Hence, a natural way to approach our problem is as one of learning a distance metric between pairs of images. We have considered several possible ways to learn such a distance metric using the image graph. For example, one could take all the connected pairs in the graph to be positive example pairs, and all other pairs as negative example pairs, and learn a single, **global** image distance metric for a specific image graph, e.g., using the machinery of SVMs [3]. At the other extreme, one could learn a **local** distance metric for each image in the database, similar to how Exemplar-SVMs have been used for object detection [22].

We tried both of these approaches, but found that, in practice, we achieve better performance with an approach that balances these two extremes. In particular, we divide the graph into a set of overlapping, representative subgraphs, and learn a separate distance metric for each of these representative subgraphs (we will also refer to each such subgraph as a "neighborhood," because we select each subgraph as the neighborhood of a particular exemplar image). Our method can thus adaptively learn the appearance of different parts of the scene, but chooses these parts effectively so as to result in a stable set of learning problems. These learned distance metrics are then used at query time to determine to which neighborhood a query image belongs. Our approach consists of the following steps:

**At Training Time**

1. Compute a covering of the graph with a set of overlapping subgraphs.
2. Learn and calibrate a distance metric for each subgraph.

**At 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 detailed matching and geometric verification with the top database images in the shortlist, until a successful true image match is found.
5. Optionally use this match to further refine the query image location, e.g., through pose estimation.

We now describe each step in more detail. Later, in Section 5, we discuss how we further improve Step 3 by reranking the shortlist to encourage diversity, based on the structure of the graph.

**Step 1: Selecting representative neighborhoods.** We begin by selecting a set of representative subgraphs that *cover* the entire graph. Once we have selected these subgraphs, we will learn a local similarity function for each subgraph, using the images in the subgraph as positive examples, and other, unrelated images in the graph as negative examples. What makes a good subgraph for learning such local distance functions? We want each subgraph to contain images that are broadly similar, so that our learning problem has a set of positive example images that are relatively compact in appearance

[1] Note that feature detectors such as SIFT often detect unstable features in an images that are not matched to features in any other image in the collection.

space, and 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, our algorithm covers 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 $\mathcal{G}$, 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. For efficiency and robustness of learning, we would like $C$ to be as small as possible, and conversely 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 [12]; here we use them to define a set of neighborhoods for which we will compute learned distance functions.

Finding an exact minimum dominating set is an NP-complete problem, but this problem has several approximation algorithms. In our case, we use a simple greedy algorithm to find an approximate solution [11]. Starting with the empty set, we iteratively add one image at a time to the set of selected exemplars, $C$. At each iteration we maintain, for each candidate image $a$, the number of as-yet uncovered images $d_a$ that $a$ covers, and choose the candidate that maximizes $d_a$ to add to the selected set. This process repeats until all images are covered. Figure 2 shows an example image graph for the Dubrovnik dataset [19] and the exemplar images selected by this greedy algorithm (shown as red nodes). Figure 3 shows some example neighborhoods found in this way, each of which is sorted by the Jaccard index compared to the center image. Note that the Jaccard index values tend to be lower than one might expect, i.e. the number of points each pair of image share is often a small fraction (e.g., 0.2) of the total number of points visible in each image. This is likely due to the noisy process of feature matching as well as the conservative outlier rejection used by structure from motion. In this way, we seek to use the image graph to give us a clean separation between sub-locations, as well as sufficient training examples for each location.

**Step 2a: Discriminative learning on neighborhoods.** For each neighborhood selected in Step 1, the next step is to learn a distance metric for comparing new images to each neighborhood. We treat this as a

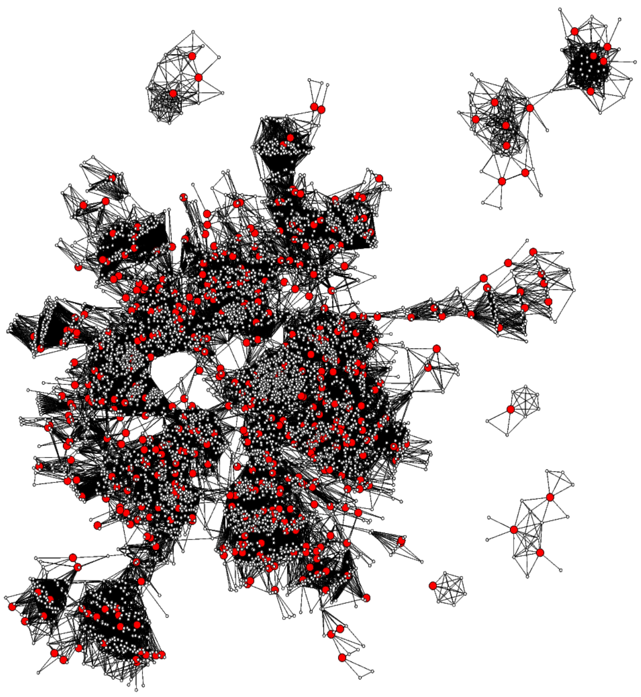
**Fig. 2 Image matching graph for the Dubrovnik dataset and selected exemplar nodes.** This graph contains 6,844 images (white dots); the large, red nodes denote representative exemplar images selected by our covering algorithm (478 images in total). Although the set of representative images is much 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. (Note that the layout of the graph is based on its structure, and not any underlying image locations.)

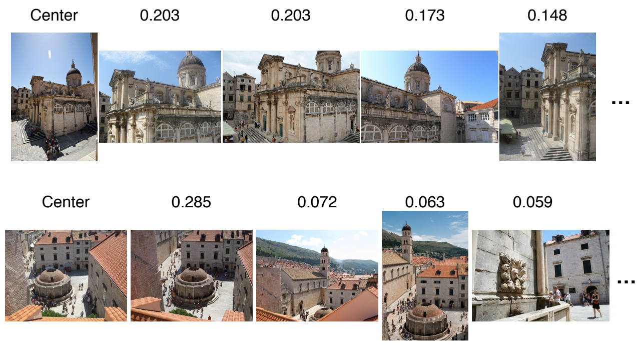

**Fig. 3 Two example neighborhoods produced by our covering algorithm.** Each row shows an example neighborhood. The leftmost image in each row is the exemplar (center) image for each neighborhood. The remaining images are sample members of that neighborhood, sorted in decreasing order by their edge weights (Jaccard index) with the center image. We have found that this way of representing locations—as collections of overlapping image neighborhoods—is a natural one, allowing us to use discriminative learning to learn the most representative features to best identify the location of a query image, adapting to each neighborhood.

classification problem, i.e., we seek to learn a classifier for each neighborhood 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 (BoW) histograms, one for each neighborhood, and calibrate the set of SVMs as described in Step 2b. At query time, these classifiers are 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 [10] or Exemplar-SVMs [31]. Note that while we use the term "exemplar" for the central image of each neighborhood, we use more than the single exemplar image as a positive example for that neighborhood; multiple images from each neighborhood are used as positives for training a classifier for recognizing new images that belong to that neighborhood.

To learn these classifiers, for each exemplar image $c \in C$ defining a neighborhood, 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 of $c$. In particular, we found that thresholding the edges in the graph by their weight—effectively applying a stricter definition of connectivity, and yielding more compact neighborhoods—yields better classifiers than using all edges found by the image matching procedure. In other words, to define the positive set for exemplar image $c$, we select all connected images $a$ such that $W(a, c) > \tau$, where $W(a, c)$ is the weight of edge $(a, c)$ (e.g., the Jaccard index $J(a, c)$), and $\tau$ is a threshold.

To define the negative set for the neighborhood around exemplar $c$, we want to minimize the chance of including false negatives in the training set. Hence we use the original graph, as opposed to the thresholded graph, to define the negatives. More specifically, we select all images $b$ such that $(b, c) \notin \mathcal{G}$ (we define $W(b, c) = 0$ in this case). In this way, the image graph $\mathcal{G}$ provides the supervision necessary to define positives and negatives for learning, just as geotags have been used as a supervisory cue for discriminative location recognition in previous work [30,16].

Given the training data for each neighborhood, we learn a linear SVM to separate images which belong to the neighborhood from images which do not. We use tf-idf weighted, $L_2$ normalized bag-of-words histograms for each image as features.[2] We randomly split the training data into training and validation subsets for parameter selection in training the SVM, and use both for fitting a logistic regressor in Step 2b (more details in Section 6.2). For each exemplar image $c$ and its neighborhood, 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 neighborhood defined by exemplar image $c$, which we will use to rank the neighborhoods. Figure 4 shows visualizations of a few results of this training procedure. Each image is overlaid with features assigned highly positive or negative weights by our learned SVM models. (Note that features are rarely assigned negative weights, but such features are potentially discriminative as usually belonging to other locations.) One interpretation of the highly positive features is that they are the ones that are both discriminative of their particular neighborhood, and consistent within their own neighborhood. The beauty of the image graph is that its structure gives us an elegant way to learn these weights automatically.

These learned weight vectors can be thought of as defining a similarity between a query and a database neighborhood as a dot product between the query and the weight vector (plus a bias term). Alternatively, at this point, we could simply *replace* our entire database with these set of weight vectors $\mathbf{w}_c$ (plus an extra dimension for the bias term). Ranking the exemplar images by classifier score would be equivalent to computing the dot product of a query image with each weight vector, and sorting by these dot products. (This can be thought of as replacing each exemplar image's original bag-of-words vector in the database with a new, discriminatively trained bag-of-words vector.) However, we first need to calibrate all of the different SVMs with respect to each other. In addition, we found it important to rank *all* database images (not just the exemplars) for a given query, as described in Step 3 below.

**Step 2b: Calibrating classifier outputs.** Since the classifier for each neighborhood is 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 [25] on the entire set of training data. We found that using more data yields better performing logistic regressor, hence unlike SVM training, we use both training and validation sets. For a neighborhood around exemplar $c$, and a query image vector $\mathbf{q}$, we denote the resulting probability value as $P_c(\mathbf{q})$. We found this independent calibration procedure to work well in practice.

[2] Note that the tf-idf weighting of the bag-of-words histograms could be seen as redundant for the purposes of learning an SVM since the SVM is itself learning a per-word weight. However, we found such "pre-weighting"—i.e., using tf-idf weighted histograms as opposed to raw histograms as feature vectors—to be useful in practice, perhaps as a form of regularization.

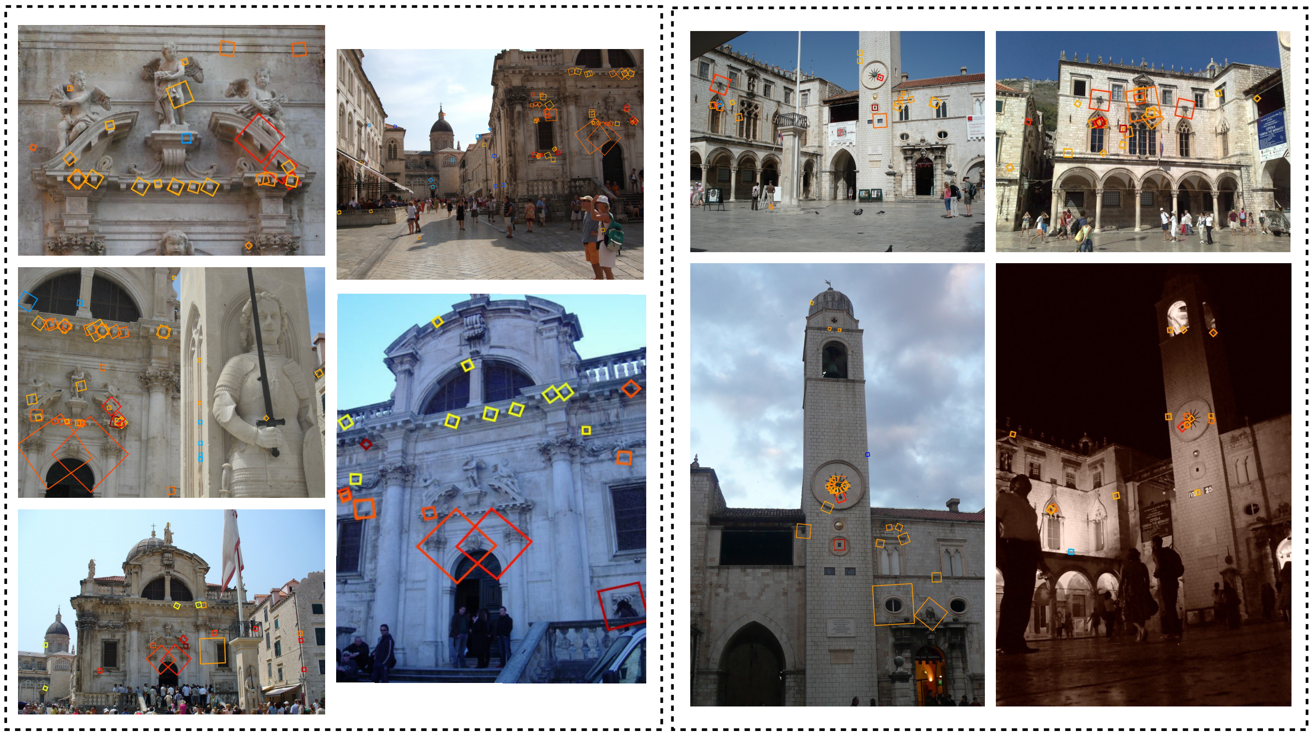

**Fig. 4 Visualization of learned SVM weights on two example neighborhoods.** Here, we show, for two neighborhoods, several example images from each neighborhood. The SIFT features shown in each image are color-coded to visualize the weights for their corresponding visual words, as learned through our SVM training process for each neighborhood. The closer to red the color, the higher the corresponding weight is; the closer to blue, the lower the weight is. Most of the features (typically $> 99\%$) are assigned zero weights (and are not shown above), due to the $L_1$ regularization used for training our SVMs and due to the large vocabulary size (1M in our experiments, see Section 6). The features shown here are assigned high positive or negative weights. Note that most features have positive weights, whereas negative weights provides better separation between different neighborhoods. For example, in the top-right image of the neighborhood on the left, negative weights on the other side of the building belong to another neighborhood while features on the front side get highly weighted. In addition, note how repetitive features get mid-range positive weights while discriminative and distinctive features get highest weights.

**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 value $P_c(\mathbf{q})$ for $\mathbf{q}$ 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 verifying the query image against exemplar images alone 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 its most highly ranked neighborhood. This results in a ranking of the entire set of database images. Note that we cannot use the learned weights directly here, since each neighborhood corresponds to a single weight vector that does not differentiate members within that neighborhood.

**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 additionally have a 3D structure-from-motion model associated with the database, we can further associate 3D points with matches in the query image, and determine the query image's camera pose [20]. Otherwise, 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.

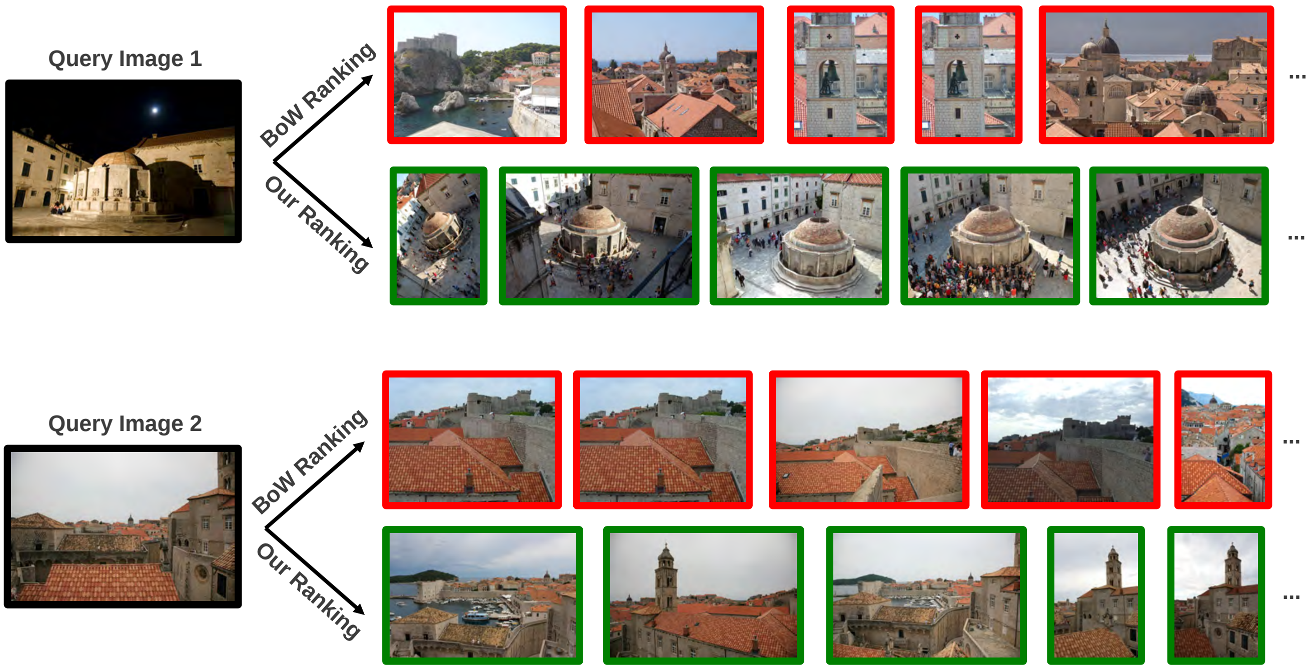

**Fig. 5 Two example query images and their top 5 ranked results using our learned similarities and raw tf-idf BoW retrieval.** For each result, a green border indicates a correct match, and a red border indicates an incorrect match. The two example query images on the left are difficult for BoW retrieval 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.

Using this simple approach, we observe improvements in our ranked lists over raw BoW retrieval results, as shown in the examples in Figure 5. In particular, the top image in the ranked list is more often correct using our ranking based on learned similarities. 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.

## 5 Diverse Shortlists

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. In addition, we demonstrate several techniques to introduce forms of regularization into our ranking to further improve recognition performance.

### 5.1 Probabilistic Reranking

In a way, our recognition problem is akin to the well-known Web search ranking problem, as compared to the standard formulation of the image retrieval problem. In our setting, rather than retrieve *all* instances relevant to a given query, it is more useful to retrieve a small set of results that are both *relevant* and *diverse* (see Figure 6 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.

In the information retrieval literature, a concept known as the *Probability Ranking Principle* captures the idea that ranking documents in decreasing order of probability of relevance provides an optimal ranking [26]. While this idea has merit if many documents are to be returned, if instead a retrieval system is to return a small number of documents, then returning the most probable set of documents may no longer be desirable. Instead, one can argue for *diversity* in such scenarios (as argued by Chen and Karger [4], among others in the information retrieval literature). In short, their argument is that in many cases a better goal is to *maximize the probability of finding a relevant doc-*

*ument among the top n*. We extend this idea to our problem setting. However, unlike Chen and Karger, who must perform this probabilistic reasoning using document features alone, we have the benefit of having the graph structure as a form of supervision for performing such probabilistic reranking. Hence, as with learning similarity functions, we leverage our image graph once more.

The idea for our probabilistic approach for reranking the shortlist is, in some ways, a sort of "anti"-query expansion. In standard image retrieval, the idea of query expansion, or other forms of positive relevance feedback, is to use successful matches early in the shortlist to move other, similar images up the ranking, which typically increases recall. In our case, we use a form of blind *negative feedback* to increase the pool of diverse matches. In particular, in anticipation of the case where the first retrieved image is *not* a match to the query, we want to precompute the probability of the second image conditioned on this outcome, likely 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? This is where we turn to the image graph.

First, let us define our notation. Suppose we have a query image $q$. 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. The calibrated probability values $P_c(\mathbf{q})$ (which in what follows we denote as $P_c$ as shorthand), computed using the algorithm in Section 4, give us an appearance-based estimate of these probabilities for each exemplar image $c$, i.e., $\Pr(X_c = 1) = P_c$. Similarly, let us define $\Pr(X_a = 1) = P_a$ for *any* database image $a$, using the simple heuristic above that the probability $P_a$ of a non-exemplar database image matching the query takes the maximum probability $P_c$ of all neighborhoods that $a$ belongs to.

To choose the top-ranked image for the query, we select the database image $a$ with the highest appearance-based probability $P_a$ (breaking ties by falling back to the BoW similarity, as in Section 4). However, to select the *second* ranked image $b$, we are instead more interested in the conditional probability $\Pr(X_b = 1|X_a = 0)$ than its raw appearance-based probability $\Pr(X_b = 1)$ alone. We denote this conditional probability as $P'_b$, which we can compute as:

$$\begin{aligned} P'_b = \Pr(X_b = 1|X_a = 0) &= \frac{\Pr(X_b = 1, X_a = 0)}{\Pr(X_a = 0)} \\ &= \frac{\Pr(X_b = 1) - \Pr(X_b = 1, X_a = 1)}{1 - \Pr(X_a = 1)} \\ &= \frac{P_b - \Pr(X_b = 1|X_a = 1)\Pr(X_a = 1)}{1 - P_a} \\ &= \frac{P_b - P_{ba}P_a}{1 - P_a} = P_b \left( \frac{1 - \frac{P_{ba}}{P_b}P_a}{1 - P_a} \right) \end{aligned} \tag{2}$$

where $P_{ba}$ is shorthand for $P(X_b = 1|X_a = 1)$, and denotes the conditional probability that image $b$ matches the query given that image $a$ matches the query. We can think of the last line in the derivation above as relating $P'_b$ to $P_b$ via an *update factor*,

$$\frac{1 - \frac{P_{ba}}{P_b}P_a}{1 - P_a} \tag{3}$$

that depends on $P_a$ (the probability that the top ranked image $a$ matches the query) and $P_{ba}$ (the conditional probability that $b$ matches given that $a$ does). How do we compute $P_{ba}$? This is where the image graph comes in. The intuition here is that the more similar $b$ is to $a$—i.e., stronger the connection between $a$ and $b$ in the image graph—the higher $P_{ba}$ should be. In particular, we define

$$P_{ba} = \frac{N(a, b)}{N(a)}, \tag{4}$$

i.e., the number of shared features between $a$ and $b$ divided by the total number of matchable feature points in $a$. We use this empirical measurement of the percentage of shared features out of all $a$'s matchable features to estimate $P_{ba}$. Note that in general $P_{ab} \not\equiv P_{ba}$, i.e., this similarity measure is asymmetric (as expected for conditional probabilities). These graph-based similarity measures are pre-computed along with the Jaccard indices $J(a, b)$ described in Section 3.

The update factor in Eq. (2) has an intuitive interpretation: roughly speaking, if image $b$ is very similar to image $a$ according to the graph (i.e., $P_{ba}$ is large relative to $P_b$), 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 slightly boosted by this update factor, as the fact that it is dissimilar to $a$ is at least weak evidence in support of $b$ if $a$ does not match the query. More precisely, the ratio $\frac{P_{ba}}{P_b}$ in the update factor depends on whether image $b$ is more similar to image $a$ or to the query image $q$, where "similarity to $a$" is measured using the image graph, and "similarity to $q$" is measured using the calibrated SVM outputs.

In practice, 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:

$$\frac{1 - \alpha \frac{P_{ba}}{P_b} P_a}{1 - \alpha P_a}. \tag{5}$$

If $\alpha = 0$, the update has no influence on the ranking result, and if $\alpha = 1$, it has its full effect. We use $\alpha = 0.9$ in our experiments.

Next, we select the third image, now based on the conditional probability that the first two images fail to match, and so on for the fourth image, etc. For selecting the third image $c$, we are interested in the conditional probability $P(X_c = 1|X_b = 0, X_a = 0)$. We make two simplifying assumptions in computing this conditional probability. First, we assume that $X_b = 0$ and $X_a = 0$ are independent events. Although we compute $b$ by maximizing a conditional probability on $a$, by construction $b$ will very often not be closely connected to $a$ in the graph; hence this conditional independence is a reasonable assumption. Second, we assume that the event "both $c$ and $b$ match query" is independent of $a$'s failure to match the query. Intuitively, this assumes that the overlapping area between image $c$ and $b$ doesn't intersect $a$, i.e. $b$ and $a$'s intersections with $c$ do not overlap. Again, this assumption is likely to hold because $b$ is chosen to be distinct from $a$ in the first place by the diversity algorithm. Hence, we have

$$\Pr(X_c = 1, X_b = 1|X_a = 0) = \Pr(X_c = 1, X_b = 1). \tag{6}$$

Given these assumptions, the update factors at each round become very simple to compute: after selecting an image, we simply replace each probability value $P_i$ for each remaining image with the updated version $P'_i$, and treat this as factoring in all previous updates. We then compute the next image via the same update factor in Eq. (5), but using these new probabilities. In other words, at each round, the evidence from all previously ranked images are integrated in an iterative fashion by repeatedly updating all candidate images' probability scores using the update factor in Eq. (5). A more complete derivation for this method can be found in the Appendix. At each iteration, we select the image for the shortlist that maximizes the modified probability at that iteration. One way to look at this method is as another kind of greedy algorithm for covering the graph, but one in which we also take into account appearance similarity to the query image.

## 5.2 Bag-of-Words Regularization

Another issue we have found to be important is that while our learned discriminative models generally perform well, for certain rare query images, our models consistently perform poorly. This is perhaps due to sparser parts of the graph having relatively few training examples, as is evident in the sparser parts of 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 have explored three ways of doing this:

- **Fall-back strategy.** First, 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 case occurs in $\sim 5\%$ of queries.)
- **Averaging of scores.** Second, to regularize our probability scores in case of overfitting in the learning process, 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.
- **Interleaving.** Third, we found that our learned models and the original tf-idf scores sometimes performed in a complementary way; 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 fall-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 we use to determine 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 in all of our experiments, and additionally evaluate a combination of averaging and interleaving as a stronger form of tf-idf regularization. Figure 7 illustrates how we combine the tf-idf scores and our own learned probabilities to form a (dynamic) ranking of the database images.

**Discussion.** One could argue that the necessity of increasing diversity in our case is caused by the decision in Step 3 in Section 4 to generate a ranked list of the entire image set rather than only the exemplar images. For instance, perhaps choosing the single image with the highest BoW similarity to the query image from

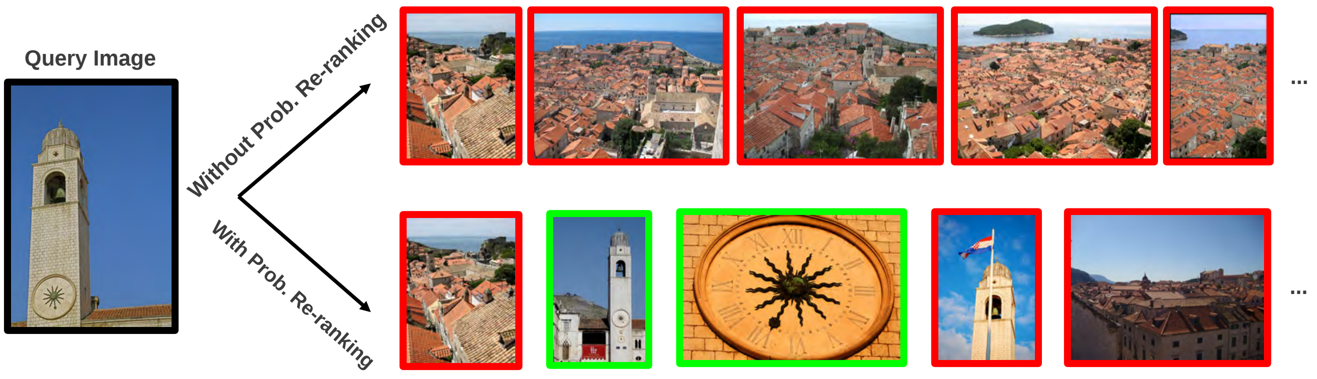

**Fig. 6 An example query image and the top 5 ranking results using our method with and without probabilistic ranking.** Green borders indicate correct matches, and red borders incorrect ones. Without probabilistic ranking, our algorithm generates a top 5 set of results that are similar, and all incorrect. With probabilistic reranking, more diversity is encouraged in the top ranking results, leading to several correct images among the top 5 results.

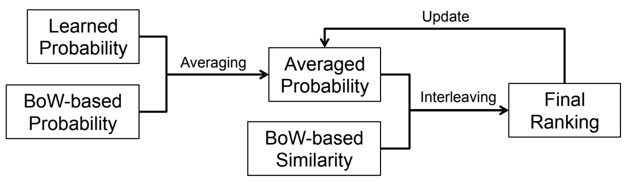

**Fig. 7 The overall procedure for our BoW-based regularization of the shortlists.** In each round of verification for the query image, we select the best scoring image (alternating between averaged probability and BoW similarity) in the **Final Ranking** step. If this selected image doesn't match the query, we update the average probabilities for the remaining images using Eq. (5).

each neighborhood might solve the problem. We have tried this approach (**GBP+MaxBoW** in Table 2) on the Dubrovnik dataset, and found that overall the diversity approach performed better. It is unclear how each neighborhood should be sampled, either by BoW similarity, or other methods, that ensures not only that query image has a high probability to be successfully matched if the query image is in that neighborhood, but also the true neighborhood doesn't appear too far down the ranking list if the query image is not.

## 6 Evaluation

In this section, we evaluate different variants of our proposed method, and compare to several baseline approaches, on a range of datasets.

As discussed in Section 4, 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 (more expensive) detailed matching and geometric verification procedure on as few images as possible. Hence, to evaluate an ordering of images in the shortlist, we use accuracy at top $k$ ($k \in \{1, 2, 5, 10\}$), i.e., the percentage of query images for which at least one correct match exists in the top $k$ retrieved results. Note that all the methods we test are compared on an equal footing, based purely on the shortlist they generate, without use of RANSAC-based verification before examining results. In all cases we apply detailed verification on each short-listed image sequentially until the first true match is found, at which point a localization is achieved.

### 6.1 Datasets and Preprocessing

We evaluated our algorithm on several location recognition datasets, including the Dubrovnik and Rome datasets [19], the Aachen dataset [29], and a much larger Landmarks dataset [20] consisting of 1,000 separate landmarks across the globe; these datasets are summarized in Table 1, along with statistics over the neighborhoods we compute for each dataset. To represent images as BoW histograms, we learned two kinds of visual vocabularies [24]: 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 the pair has at least 12 inlier feature matches after pairwise matching and geometric verification.

**Table 1 Summary of datasets used in our experiments, along with their computed neighborhoods.** Each row summarizes a dataset, showing the number of query images in the test set, the total number of images in the database, and the number of neighborhoods our algorithm selects in order to cover each image graph. The representative neighborhoods (clusters) are found using graphs whose edge weights are defined using the Jaccard index, and thresholded using a value of 0.01. The rightmost column shows the average cluster size in each dataset, along with the standard deviation in the cluster sizes.

| Dataset | # Queries | # DB Images | # Clusters | Avg. Cluster Size |
|---|---|---|---|---|
| Dubrovnik [19] | 800 | 6,044 | 188 | 103.9 ($\pm$103.4) |
| Rome [19] | 1,000 | 15,179 | 352 | 146.5 ($\pm$ 180.8) |
| Aachen [29] | 369 | 4,479 | 161 | 41.2 ($\pm$ 37.8) |
| Landmarks [20] | 10,000 | 206,162 | 8,803 | 67.3 ($\pm$ 102.7) |

## 6.2 Evaluation Methodology

For each dataset, we compare several algorithms for recognizing each query image:

(a) **BoW**: Standard tf-idf weighted bag-of-words (BoW) image retrieval [24].
(b) **BoW+RR**: A probabilistic reranked version of (a) using our method described in Section 5.1, and the calibrated BoW similarities as probabilities. To calibrate, we randomly select equal number of matching and non-matching image pairs from the database and fit a logistic regression model using the BoW similarities of these pairs as the input feature. This allows us to evaluate reranking independent of our learning method.
(c) **GBP**: ("Graph-based probability.") Our graph neighborhood based similarity learning technique (Section 4).
(d) **GBP+RR**: Our learning method using probabilistic reranking (Section 5.1).
(e) **GBP+RR+BoW**: Our method with both probabilistic reranking and the BoW regularization (Section 5.2).

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 [6] and a baseline that uses geotags on images as a supervisory signal, inspired by work on learning confusing features [30,16]. In particular, for this last baseline, we randomly select a set of exemplar images, define the nearest neighbors using GPS positions (we use 200 nearest neighbors in our experiments) as positives and the remaining images as negatives and then use the same learning and retrieval techniques described above using these neighborhoods. In addition, on the Dubrovnik dataset, 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 an exemplar (i.e., learning a per-image distance metric). Finally, also on the Dubrovnik dataset, we evaluate an alternative approach to combine BoW similarity and our GBP score (as metioned in the discussion towards the end of Section 5.2), i.e. choose one image from each neighborhood with the maximum BoW similarity using the order determined by GBP score.

Note that while we use a relatively simple image representation (weighted BoW histograms), our method is orthogonal to many other improvements to bag-of-words models [2], and can generalize to more sophisticated feature representations.

**Experimental details.** We construct a Jaccard-index weighted image graph $\mathcal{G}$, and threshold by $\tau = 0.01$ to obtain a set of positive image pairs. From the graph $\mathcal{G}$, we choose exemplar images (neighborhoods) and learn $L_1$-regularized linear SVMs and logistic functions as described in Section 4. We found that $L_1$ regularized SVMs perform slightly better than $L_2$ regularized SVMs for our problem, and the $L_1$ models also have the advantage of yielding much sparser weight vectors (typically $< 1\%$ non-zeros compared to typically $> 99\%$ non-zeros for $L_2$ regularized models) and hence faster query times. For each cluster, we use all the available positive examples (i.e., cluster sizes in Table 1), and sample roughly 5 times as many negative examples as positives. For each learning problem, one-third of the training data is held out at random for validation, and all training data is used for logistic regressor training. We use *liblinear* [8] for SVM and logistic regressor training. For each query image, we compute the estimated probability of it matching each cluster, and obtain the initial ranking of the database images as described in Section 4. We show the results of our method (a) ranking with just the graph-based probability scores (**GBP**), (b) ranking neighborhoods using graph-based scores then choose the image in each one of them with the maximum BoW similarity in the ranked order (**GBP+MaxBoW**), (c) reranking using our diversity measure (**GBP+RR**), and (d) the stronger form of BoW regularization (**GBP + RR + BoW**) using a weighted average of the two probability scores, with a weight of 5/6 on our GBP score, and 1/6 on the tf-idf-based probability score, as well as an interleaving ranking (described in Section 5).

## 6.3 Runtime analysis

Our approach involves additional overhead at training time, as well as a small amount of extra work at runtime. Suppose we have an image set of size $S$, and each image is encoded in an $n$-dimensional BoW vector. $\sigma$ is the average ratio of non-zero elements in the BoW vectors, and $m$ is the number of neighborhoods selected by our algorithm. Our greedy algorithm for selecting neighborhoods takes time is $\mathcal{O}(mS)$, as in each iteration we consider at most $S$ candidate images. Training the linear SVMs tanks time $\mathcal{O}(mSn\sigma)$, since the total number of training examples (both positives and negatives) is $\mathcal{O}(S)$. Note that SVM training is also faster with stronger regularization.

At runtime, for a query, we first need to compare the BoW vector of a query image to the $m$ learned weight vectors, then compare it against all images for BoW regularization, hence the running time for computing these similarities is $\mathcal{O}((m+S)n\sigma)$. To generate the diverse shortlists of length $k$, each time when we choose to place an image on the ranking list, we need to recompute probabilities from $S$ candidates and find the maximum one. Hence, the running time of these operations is $\mathcal{O}(kS)$. In summary, to generate a length-$k$ shortlist for a query image, the total running time is $\mathcal{O}((m+S)n\sigma+kS)$.

To put these variables into perspective in practice, for Dubrovnik dataset, there are $S = 6044$ images and $m = 188$ neighborhoods (Table 1), and the average non-zero ratio for BoW vectors is $\sigma = 2.5\%$. To give some examples measured training and querying running times of our approach, the average training time (SVM training and calibration) is about 40 seconds per model, and the average query time is about 5 seconds per query for $k = 10$. In addition, since we need to store an additional weight vector for each neighborhood, the memory usage of our algorithm has the additional overhead of storing $m$ weight vectors (of size $\mathcal{O}(mn\sigma)$) compared to standard image retrieval methods.

## 6.4 Results

The results of all experiments are shown in Table 2. We show results using both specific and generic vocabularies for the Dubrovnik dataset, and only using the specific vocabulary for the other datasets, as we found

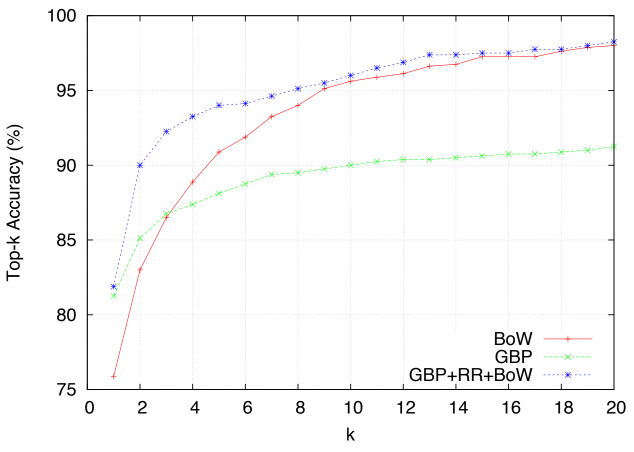

**Fig. 8 Top-k accuracy on the Dubrovnik dataset (generic vocabulary).**

that across the datasets the specific vocabulary outperformed the generic vocabularies. (We were interested in seeing whether our learning methods could overcome the disadvantages of the generic vocabulary, but found that there was still a significant advantage to specific vocabularies even with learning.)

**Graph-based methods.** A few trends are evident from the results. Our graph-based probability method (**GBP**), by itself, consistently improve results for the top1 and top2 rankings over the baseline BoW method; the improvement in top1 success rate ranges from a small amount for the Rome dataset, to nearly $> 7\%$ for the Dubrovnik dataset (with specific vocabulary); the average improvement is 3.8% across all datasets. However, the performance of the GBP method increases more slowly than the baseline tf-idf ranking as a function of $k$, and for the top5 and top10 rankings GBP performs worse in many cases. However, once we reintroduce diversity through probabilistic reranking (**RR**), our results improve on average for these longer shortlists (3.9% on average across our datasets for top10). We observe small additional gains when regularizing our learned results with the tf-idf BoW scores. Figure 8 shows the top-$k$ accuracy performance on the Dubrovnik dataset (trained with a generic vocabulary) as a function of $k$. Although the GBP performance alone is worse than our BoW baseline for larger $k$, probabilistic reranking and BoW regularization improve the performance to be better than the baselines.

Note that there is a significant variety in the structure and difficulty of our datasets. Rome is a relatively easy dataset, consisting of many separate scenes (most of which are small and compact), while Dubrovnik has a more interesting graph structure, consisting of a single large connected component spanning many view-

**Table 2 Recognition performance on all datasets.** The abbreviations for each method are defined in Section 6.2.

Dubrovnik (Specific Vocab.)

| Method | top1 | top2 | top5 | top10 | mAP |
|---|---|---|---|---|---|
| BoW [32] | 87.50% | 92.75% | 97.62% | 98.50% | 0.401 |
| BoW+RR | 87.50% | 93.38% | 96.63% | 97.50% | 0.058 |
| Co-ocset [6] | 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+MaxBoW | **94.38%** | **97.50%** | 99.00% | 99.13% | 0.200 |
| 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 |

Landmarks

| Method | top1 | top2 | top5 | top10 | mAP |
|---|---|---|---|---|---|
| BoW | 58.42% | 64.10% | 71.22% | 75.81% | 0.115 |
| BoW+RR | 58.42% | 65.97% | 77.04% | 80.76% | 0.064 |
| GBP | 62.58% | 64.59% | 66.90% | 68.35% | **0.471** |
| GBP+RR | 62.58% | 65.19% | 75.34% | **81.17%** | 0.084 |
| GBP+RR+BoW | **64.60%** | **73.03%** | **78.01%** | 81.01% | 0.052 |

points across a city. Landmarks consists of the union of many separate 3D models, some large, some small, and is the most difficult dataset due to its large size. For the Landmarks dataset, the improvements we see using our method are particularly large, over 5.36% for top10 over the bag-of-words baseline, with the diversity reranking being the most significant factor in this improvement. This suggests that our method may be particularly effective for improving recognition performance with large datasets. We found that adding diversity to the raw BoW method (i.e., **BoW+RR**) did not consistently perform better than the standard BoW method. However, for the Landmarks dataset diversity significantly improves the BoW method (from 76% to 81% for top10) just as it does our GBP method.

**Other baseline methods.** In the Dubrovnik (Specific Vocab.) results, our full method outperformed each of the unsupervised baseline methods (BoW, BoW+RR and Co-ocset [6]). As described above, we also evaluated several supervised baselines. The GPS-based model (**GPS Model**) did not improve over the unsupervised baselines (in fact, did worse in general). This is likely due to the less careful choices of training examples compared to those based on image graph. The globally trained similarity metric (**Global Model**) also performed 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 significantly improve the mAP (mean average precision) score (computed on a ranking of all images), suggesting that this method is much better at globally ranking the images than it is at our recognition task—in other words, it is doing better at the problem of retrieving all relevant instances. The per-image classifiers (**Instance Model**) perform best among the baselines, but the GBP method still performs better. 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 can 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). Compared to our proposed BoW regularization method (+BoW), choosing the image with the maximum BoW similarity in the ranked order by GBP scores (+MaxBoW) performs better for top1 and top2 images, but worse for top5 and top10.

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 [20] (100% / 99.7%), which uses direct 3D matching and requires much more memory and expensive nearest neighbor computations. This effort is needed since these methods store entire 3D point clouds (often with millions of individual points), as well as a 128-byte SIFT feature for each point—often requiring many gigabytes of memory. In our case, we only store a set of quantized features, BoW image vectors, and sparse weight vectors. For example, for the Dubrovnik dataset, our unoptimized Python implementation of our system uses about half the memory as the C++ implementation of [20]. Our performance on Aachen dataset (89.16%) also rivals that of [29], 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.

## 7 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 types of 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 National Science Foundation (grants IIS-1149393 and IIS-0964027) and Intel Corporation. We also thank Flickr users for use of their photos.

## References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S., Szeliski, R.: Building Rome in a day. In: ICCV (2009)
2. Arandjelovic, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: CVPR (2012)
3. Cao, S., Snavely, N.: Learning to match images in large-scale collections. In: ECCV Workshop on Web-scale Vision and Social Media (2012)
4. Chen, H., Karger, D.R.: Less is more: probabilistic models for retrieving fewer relevant documents. In: ACM SIGIR (2006)
5. Chum, O., Matas, J.: Large-scale discovery of spatially related images. PAMI (2010)
6. Chum, O., Matas, J.: Unsupervised discovery of co-occurrence in sparse high dimensional data. In: CVPR (2010)
7. Doersch, C., Singh, S., Gupta, A., Sivic, J., Efros, A.A.: What makes Paris look like Paris? SIGGRAPH (2012)
8. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: Liblinear: A library for large linear classification. The Journal of Machine Learning Research (2008)
9. Frahm, J.M., et al.: Building Rome on a cloudless day. In: ECCV (2010)
10. Frome, A., Singer, Y., Sha, F., Malik, J.: Learning globally-consistent local distance functions for shape-based image retrieval and classification. In: ICCV (2007)

11. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. Algorithmica (1998)
12. Havlena, M., Torii, A., Pajdla, T.: Efficient structure from motion by graph optimization. In: ECCV (2010)
13. Hays, J., Efros, A.: Im2gps: estimating geographic information from a single image. In: CVPR (2008)
14. Irschara, A., Zach, C., Frahm, J., Bischof, H.: From structure-from-motion point clouds to fast location recognition. In: CVPR (2009)
15. Johns, E., Yang, G.: From images to scenes: Compressing an image cluster into a single scene model for place recognition. In: ICCV (2011)
16. Knopp, J., Sivic, J., Pajdla, T.: Avoiding confusing features in place recognition. In: ECCV (2010)
17. Li, X., Wu, C., Zach, C., Lazebnik, S., Frahm, J.: Modeling and recognition of landmark image collections using iconic scene graphs. In: ECCV (2008)
18. Li, Y., Crandall, D., Huttenlocher, D.: Landmark classification in large-scale image collections. In: ICCV (2009)
19. Li, Y., Snavely, N., Huttenlocher, D.: Location recognition using prioritized feature matching. In: ECCV (2010)
20. Li, Y., Snavely, N., Huttenlocher, D., Fua, P.: Worldwide pose estimation using 3d point clouds. In: ECCV (2012)
21. Malisiewicz, T., Efros, A.: Beyond categories: The visual memex model for reasoning about object relationships. NIPS (2009)
22. Malisiewicz, T., Gupta, A., Efros, A.A.: Ensemble of exemplar-SVMs for object detection and beyond. In: ICCV (2011)
23. Mikulik, A., Perdoch, M., Chum, O., Matas, J.: Learning a fine vocabulary. In: ECCV (2010)
24. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR (2007)
25. Platt, J., et al.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in Large Margin Classifiers (1999)
26. Robertson, S.E.: Readings in information retrieval. chap. The Probability Ranking Principle in IR, pp. 281–286 (1997)
27. Sattler, T., Leibe, B., Kobbelt, L.: Fast image-based localization using direct 2D-to-3D matching. In: ICCV (2011)
28. Sattler, T., Leibe, B., Kobbelt, L.: Improving image-based localization by active correspondence search. In: ECCV (2012)
29. Sattler, T., Weyand, T., Leibe, B., Kobbelt, L.: Image retrieval for image-based localization revisited. In: BMVC (2012)
30. Schindler, G., Brown, M., Szeliski, R.: City-scale location recognition. In: CVPR (2007)
31. Shrivastava, A., Malisiewicz, T., Gupta, A., Efros, A.A.: Data-driven visual similarity for cross-domain image matching. SIGGRAPH ASIA (2011)
32. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: ICCV (2003)
33. Torii, A., Sivic, J., Pajdla, T.: Visual localization by linear combination of image descriptors. In: ICCV Workshops (2011)
34. Turcot, P., Lowe, D.: 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)
35. Zheng, Y.T., et al.: Tour the world: building a web-scale landmark recognition engine. In: CVPR (2009)

## Appendix A Derivation for the iterative usage of the update factor

Here we provide the full derivation of how our diversity reranking method updates the probability scores for each image each time we select a new image for the shortlist. Let $n$ denote the number of images that have already been selected for the ranked shortlist $RL = \{i_1, i_2, ..., i_n\}$. For any image $u$ in the as-yet-unselected set of images $\mathcal{I} \setminus RL$, we wish to compute

$$P_u^{(n)} = \Pr(X_u = 1 | X_{i_1} = 0, ..., X_{i_n} = 0),$$

which is the conditional probability of image $u$ matching the query image given that all the $n$ images in the current ranking list do **not** match the query. To simplify the analysis, we separately consider the cases $n = 1$, $n = 2$, and the general case $n = k$ (note that in Section 5 we denote $P_u^{(1)}$ as $P'_u$).

- $n = 1$: From Eq. (2), Section 5 derives the update factor in Eq. (3) used to compute $P_i^{(1)}$:

$$P_u^{(1)} = P_u \frac{1 - \frac{P_{u,i_1}}{P_u} P_{i_1}}{1 - P_{i_1}}$$

where $P_u$ is the original probability estimate from our GBP approach.
- $n = 2$: From the definition of conditional probability, we have that

$$\begin{aligned} P_u^{(2)} &= \Pr(X_u = 1 | X_{i_2} = 0, X_{i_1} = 0) \\ &= \frac{\Pr(X_u = 1, X_{i_2} = 0, X_{i_1} = 0)}{\Pr(X_{i_2} = 0, X_{i_1} = 0)} \\ &= \frac{\Pr(X_u = 1, X_{i_2} = 0 | X_{i_1} = 0)\Pr(X_{i_1} = 0)}{\Pr(X_{i_2} = 0, X_{i_1} = 0)} \end{aligned} \tag{7}$$

Recall from Section 5 that we make two simplifying assumptions:
1. *Independence of* $(X_{i_1} = 0)$ *and* $(X_{i_2} = 0)$ *for distant* $i_1$ *and* $i_2$:

$$\Pr(X_{i_2} = 0, X_{i_1} = 0) = \Pr(X_{i_2} = 0)\Pr(X_{i_1} = 0)$$

2. *Independence of* $(X_u = 1, X_{i_2} = 1)$ *and* $(X_{i_1} = 0)$:

$$\Pr(X_u = 1, X_{i_2} = 1 | X_{i_1} = 0) = \Pr(X_u = 1, X_{i_2} = 1)$$

Given these assumptions, we can simplify Eq. (7) as follows:

$$\begin{aligned}
P_u^{(2)} &= \Pr(X_u = 1 | X_{i_2} = 0, X_{i_1} = 0) \\
&= \frac{\Pr(X_u = 1, X_{i_2} = 0 | X_{i_1} = 0) \Pr(X_{i_1} = 0)}{\Pr(X_{i_2} = 0) \Pr(X_{i_1} = 0)} \\
&= \frac{\Pr(X_u = 1, X_{i_2} = 0 | X_{i_1} = 0)}{\Pr(X_{i_2} = 0)} \\
&= \frac{\Pr(X_u = 1 | X_{i_1} = 0) - \Pr(X_u = 1, X_{i_2} = 1 | X_{i_1} = 0)}{1 - \Pr(X_{i_2} = 1)} \\
&= \frac{\Pr(X_u = 1 | X_{i_1} = 0) - \Pr(X_u = 1, X_{i_2} = 1)}{1 - \Pr(X_{i_2} = 1)} \\
&= \frac{P_u^{(1)} - \Pr(X_u = 1 | X_{i_2} = 1) \Pr(X_{i_2} = 1)}{1 - \Pr(X_{i_2} = 1)} \\
&= \frac{P_u^{(1)} - P_{u,i_2} P_{i_2}}{1 - P_{i_2}} = P_u^{(1)} \left( \frac{1 - \frac{P_{u,i_2}}{P_u^{(1)}} P_{i_2}}{1 - P_{i_2}} \right) \qquad (8)
\end{aligned}$$

where $P_u^{(1)}$ is the updated conditional probability of $X_u = 1$ using the evidence $X_{i_1} = 0$. Note that the update factor at the end of Eq. (8) is of the same form of Eq. (3).

- $n = k$: More generally, we can similarly derive that

$$P_u^{(k)} = P_u^{(k-1)} \left( \frac{1 - \frac{P_{u,i_k}}{P_u^{(k-1)}} P_{i_k}}{1 - P_{i_k}} \right) \qquad (9)$$

Hence, through Eq. (9), we can update each as-yet-unchosen image $u$ conditional probability $P_u^{(k)}$ in a iterative fashion each time we add a new image to the ranking list $RL$.