City-Scale Location Recognition

Grant Schindler
Georgia Institute of Technology
schindler@cc.gatech.edu

Matthew Brown Richard Szeliski
Microsoft Research, Redmond, WA
{brown,szeliski}@microsoft.com

Abstract

We look at the problem of location recognition in a large image dataset using a vocabulary tree. This entails finding the location of a query image in a large dataset containing 3×10^4 streetside images of a city. We investigate how the traditional invariant feature matching approach falls down as the size of the database grows. In particular we show that by carefully selecting the vocabulary using the most informative features, retrieval performance is significantly improved, allowing us to increase the number of database images by a factor of 10. We also introduce a generalization of the traditional vocabulary tree search algorithm which improves performance by effectively increasing the branching factor of a fixed vocabulary tree.

1. Introduction

The existence of large-scale image databases of the world opens up the possibility of recognizing one's location by simply taking a photo of the nearest street corner or store-front and finding the most similar image in a database.

When this database consists of millions of images of the world, the problem of efficiently searching for a matching image becomes difficult. The standard approach to image-matching – to convert each image to a set of scale- and rotation-invariant feature points – runs into storage-space and search-time problems when dealing with tens of millions of feature points.

We adopt a vocabulary tree [9] to organize and search these millions of feature descriptors without explicitly storing the descriptors themselves, a method previously used in object recognition. Using vocabulary trees in the context of location recognition allows us to determine which features are the most informative about a given location and to structure the tree based on these informative features. We also present a generalization of the traditional search algorithm for vocabulary trees that allows us to improve the performance of the vocabulary tree at search time as well as during construction of the tree.

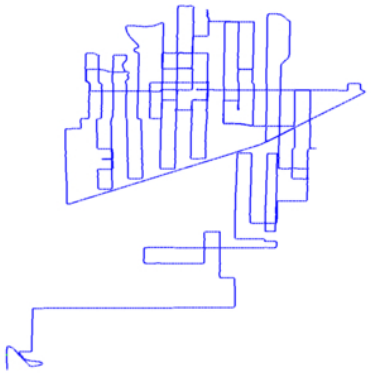


Figure 1. We perform location recognition on 20 km of urban streetside imagery, storing 100 million features in a vocabulary tree, the structure of which is determined by the features that are most informative about each location. Shown here is the path of our vehicle over 20 km of urban terrain.

1.1. Related Work

Vision-based location systems [10, 11, 14] have generally focused on small databases (200-1000 images) taken by human photographers. For example, [10] uses a database of 200 images, each of a different building facade, and requires manual identification of lines in the database images. We demonstrate results on an automatically captured 30,000 image database, consisting of over 100 million SIFT features [7], and covering a continuous 20 kilometer stretch of roads through commercial, residential, and industrial areas.

Searching for images in large databases was the focus of [12] and later [9], which used a vocabulary tree of a kind described in [2], and earlier in [3], for efficient search. In [9], images are documents described by frequencies of visual words in a latent semantic analysis (LSA) framework, where visual words are defined by both leaf-nodes and higher-level nodes in the tree, in a hierarchical fashion. We use the same basic data structure of [9], but in several fundamentally different ways. We use neither LSA, nor hierarchical scoring, and we focus on building trees for specific databases to be used for location recognition, rather than generic trees for object recognition.

The idea of choosing a reduced set of informative fea-

1-4244-1180-7/07/$25.00 ©2007 IEEE

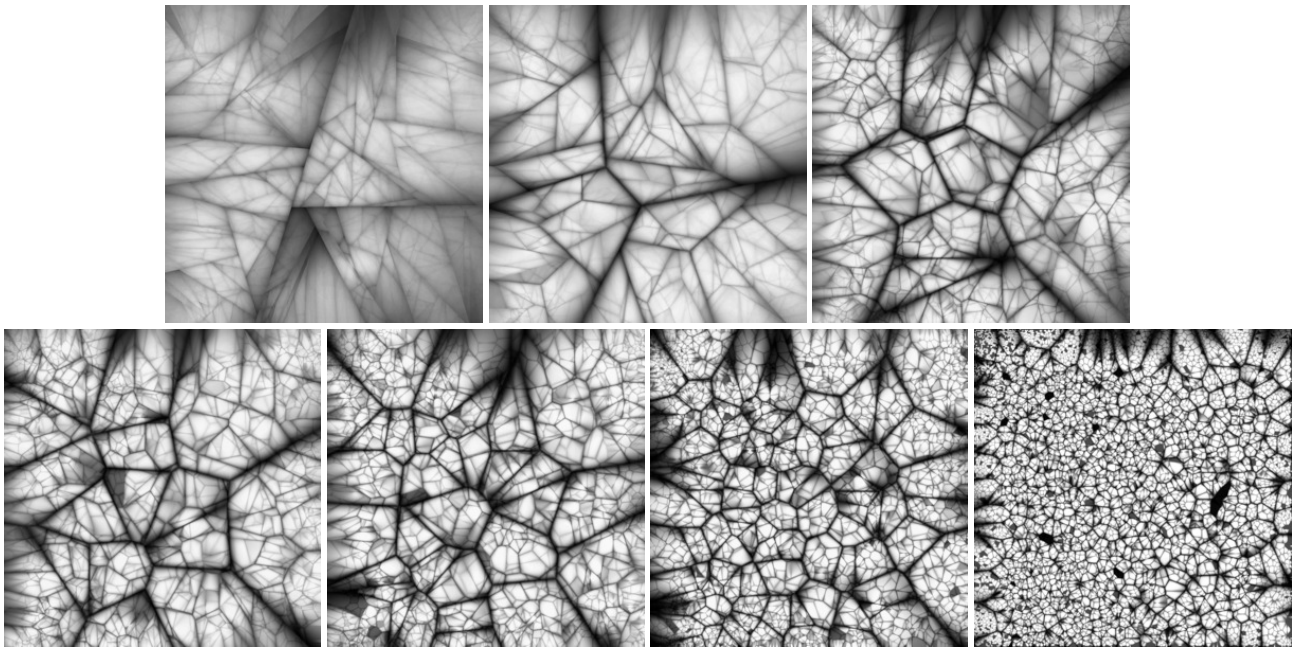


Figure 2. Vocabulary Trees of Varying Branching Factor and Depth. For branching factor k and depth L, we can produce seven trees with $k^L \approx 1,000,000$ leaf nodes. Starting from top left, the trees are of size 2^{20}, 4^{10}, 10^6, 16^5, 32^4, 100^3, and 1000^2. These nested Voronoi diagrams show each tree projected onto two random dimensions of the 128-dimensional SIFT space. Gray values indicate the ratio of the distances to the nearest and second-nearest node at each level of the tree (black=1).

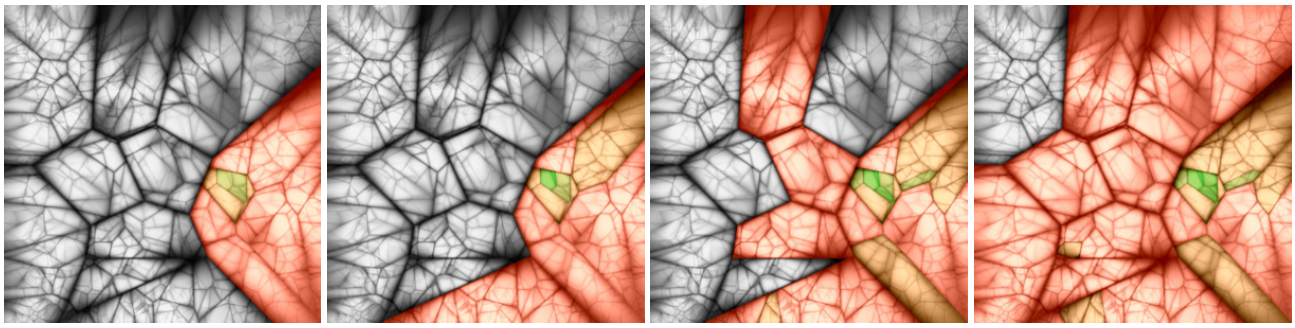
tures [13, 6] has previously been used in location and object recognition tasks. We show how to exploit the natural structure of vocabulary trees to define a feature's information. In contrast to previous work, rather than reducing the size of the feature database, we propose using feature information to guide the building of the vocabulary tree instead. The set of features stored in the leaf nodes of the resulting vocabulary tree remains the same. See [5] for a similar approach, using information loss minimization to build nearest neighbor quantizers for use as codebooks in bag-of-features image classification.

2. Vocabulary Trees

The vocabulary tree [9] is an effective data structure for searching a large database in high-dimensional spaces. As image databases increase in size, we run into several barriers that prevent traditional feature matching techniques from scaling up. For example, we may have too many feature descriptors to store in memory (30,000 images $\approx$ 100,000,000 SIFT features $\approx$ 12 GB) and far too many features to exhaustively compare against a query feature. Even kd-trees fail to solve the storage problem since approximate k-nearest neighbor algorithms like Best Bin First [1] require access to feature descriptors at search time. The vocabulary tree solves this storage problem by throwing away the descriptor for each database feature.

A vocabulary tree is a k-way tree of depth L such that there are k^L leaf nodes, or visual words, at the bottom of the tree. It is best understood as the product of a hierarchical k-means clustering of a number of descriptors, where each node in the tree is a feature descriptor equal to the mean of all the descriptors assigned to it. The tree is queried by comparing a query SIFT feature to all k nodes at the root level to find the closest match, and then recursively comparing it to all k children of that node. The database features corresponding to any visual word will all cluster around a single feature descriptor, and thus we can throw away the SIFT descriptor for all features in the database and instead store a 6-byte pointer to each feature in the form of an image number (4-byte int) and feature number (2-byte short).

Note that the vocabulary tree is an instance of a *metric tree* in the sense used by [8] to distinguish between two classes of tree-based data structures: trees which organize data in a vector space in which individual dimensions must be accessed (e.g. kd-trees), and trees which organize data in a metric space in which a metric distance between any two points can be computed. In querying a vocabulary tree, we only ever care about the distance between a query feature and each node in the tree. Thus, the structure of the vocabulary tree in the 128-dimensional SIFT space can be visualized as a nested set of Voronoi cells as in Figure 2. Trees are constructed with hierarchical k-means as described in [9], where we use Gonzalez's algorithm [4] to initialize the cluster centers with points that are as far apart from each other as possible.



(a) N=1 (b) N=2 (c) N=5 (d) N=9

Figure 3. Greedy N-Best Paths Search. From left to right, we increase the number of nodes N whose children are considered at each level of the tree. Cells are colored from red to green according to the depth at which they are encountered in the tree, while gray cells are never searched. By considering more nodes in the tree, recognition performance is improved at a computational cost that varies with N.

3. Greedy N-Best Paths Search

A popular search heuristic for approximate nearest neighbors in kd-trees is the Best Bin First (BBF) algorithm [1]. Bounds are computed on the nearest possible feature residing in each path not followed as the search descends down the tree, and a specified number of candidate features are considered in succession. We propose an algorithm similar in spirit to BBF which exploits the unique properties of metric trees to allow us to specify how much computation takes place during nearest neighbor search. We propose the Greedy N-Best Paths (GNP) algorithm, which follows multiple branches at each level rather than just the branch whose parent is closest to the query feature. This generalization of the traditional vocabulary tree search method is described in Algorithm 1.

For branching factor k and depth L, the normal search algorithm for a metric tree performs k comparisons between the query feature and the nodes of the tree at each of L levels for a total of kL comparisons. Our algorithm performs k comparisons at the top level, and kN comparisons at each of the remaining $L-1$ levels, for a total of $k+kN(L-1)$ comparisons. This allows us to specify the amount of computation per search by varying the number of paths followed N. Note that traditional search is just the specific case in which $N=1$.

Algorithm 1 Greedy N-Best Paths

Given query feature q, and level $\ell = 1$
Compute distance from q to all k children of root node
While $(\ell < L)$\{
  $\ell = \ell + 1$
  Candidates=children of closest N nodes at level $\ell - 1$
  Compute distance from q to all kN candidates
\}
Return all features quantized under closest candidate

3.1. Branching Factor

For a fixed vocabulary size M, corresponding to the number of leaf nodes in a tree, there are several ways to construct a vocabulary tree. This is accomplished by varying the branching factor k and depth L of the tree such that $k^L \approx M$ for integer values of k and L. In previous work on vocabulary trees[9], it was noted that increasing the branching factor for fixed vocabulary size tended to improve the quality of search results. We claim that much of this improvement is due not to the fact that increasing branching factor produces better-structured trees, but to the fact that more nodes are being considered in traversing a tree with higher branching factor. As an example, using a 1 million word vocabulary, consider that in a 10^6 tree only 60 nodes are ever examined while in a 1000^2 tree 2000 nodes are considered during a traditional search. The GNP algorithm offers a way to consider 2010 nodes in a 10^6 tree with $N=40$, and we show in Section 6 that comparable performance is achieved with GNP on a tree with fewer branches.

Note that changing the branching factor of a vocabulary tree requires time-consuming offline re-training via hierarchical k-means. However, varying the number of nodes searched is a decision that can be made at search time based on available computational power. Thus, we should concentrate not on the relationship between performance and branching factor, but between performance and number of comparisons per query feature, a measure which GNP allows us to optimize (see Figure 6).

4. Informative Features

One of the goals of [9] was to show that acceptable recognition performance is possible using a generic vocabulary tree trained on data unrelated to the images eventually used to fill the database. This is important when the database is expected to change on the fly. However, if the database consists of a fixed set of images, we should instead aim to build the vocabulary tree which maximizes perfor-

mance of queries on the database.

For a fixed vocabulary of 1 million visual words, we can not only vary the branching factor and depth of the tree, but also choose training data such that the capacity of the tree is spent modeling the parts of SIFT space occupied by those features which are most informative about the locations of the database images. This becomes even more important when the database becomes so large that the hierarchical k-means process used to build the vocabulary tree cannot possibly cluster all the data at once, but must instead build a tree based on some subset of the database. In selecting the subset of data for training, we can either uniformly sample the database, or choose those features which are most informative, which we explain here.

4.1. Information Gain

The images in our database contain considerable overlap, such that we end up with a number of images of each location from slightly different viewpoints (see Figure 8). Intuitively, we want to find features which occur in all images of some specific location, but rarely or never occur anywhere outside of that single location. This intuitive concept is captured well by the formal concept of *information gain*.

Information gain $I(X|Y)$ is a measure of how much uncertainty is removed from a distribution given some specific additional knowledge, and it is defined with respect to the entropy $H(X)$ and conditional entropy $H(X|Y)$ of distributions $P(X)$ and $P(X|Y)$. By definition:

$$H(X) = -\sum_x P(X = x) \log[P(X = x)] \quad (1)$$

$$H(X|Y) = \sum_y P(Y = y) H(X|Y = y) \quad (2)$$

$$I(X|Y) = H(X) - H(X|Y) \quad (3)$$

In our case, information gain $I(L_i|W_j)$ is always computed with respect to a specific location ℓ_i and a specific visual word w_j. L_i is a binary variable that is true when we are at location ℓ_i, and W_j is a binary variable that is true when the visual word w_j is in view (i.e., one of the images at location ℓ_i contains a feature which falls under visual word w_j when quantized according to the vocabulary tree). Thus, the information gain of visual word w_j at location ℓ_i, as defined in Equation 3, is:

$$I(L_i|W_j) = H(L_i) - H(L_i|W_j) \quad (4)$$

Remember that we are interested in finding those visual words at location ℓ_i that maximize this information gain value. Since the entropy $H(L_i)$ is constant across all visual words at location ℓ_i, then according to Equation 4, the visual word that maximizes the information gain $I(L_i|W_j)$ also minimizes the conditional entropy $H(L_i|W_j)$.

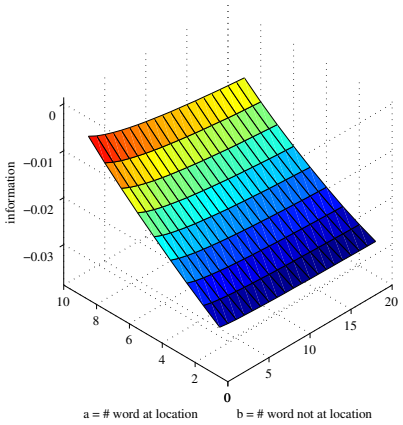


Figure 4. Information gain measures how informative a visual word is about a specific location ℓ_i, and it is computed as a function of a and b, the number of occurrences of a visual word at location ℓ_i and at all other locations, respectively. The graph shows that information gain is maximized when a visual word occurs often at the location ℓ_i and rarely at any other location.

We can calculate this conditional entropy $H(L_i|W_j)$ as a function of just four terms: N_{DB}, N_L, $N_{W_j L_i}$, and $N_{W_j \overline{L_i}}$. The first two terms are constant for a given database: N_{DB} is the number of images in the database, and N_L is the number of images at each location. The last two terms vary with each location and visual word: $N_{W_j L_i}$ is the number of times visual word w_j occurs at location ℓ_i, and $N_{W_j \overline{L_i}}$ is the number of times visual word w_j occurs at other database locations. For clarity, we substitute the variables a and b for $N_{W_j L_i}$ and $N_{W_j \overline{L_i}}$ in what follows. Note that $H(L_i|W_j)$ depends upon just six probabilities whose values follow trivially from the definitions of N_{DB}, N_L, a, and b:

$$P(L_i|W_j) = \frac{a}{a+b} \qquad P(L_i|\overline{W_j}) = \frac{N_L - a}{N_{DB} - a - b}$$

$$P(\overline{L_i}|W_j) = \frac{b}{a+b} \qquad P(\overline{L_i}|\overline{W_j}) = \frac{N_{DB} - N_L - b}{N_{DB} - a - b}$$

$$P(W_j) = \frac{a+b}{N_{DB}} \qquad P(\overline{W_j}) = \frac{N_{DB} - a - b}{N_{DB}}$$

Substituting these probabilities into Equation 2 above, we arrive at the conditional entropy

$$\begin{aligned} &H(L_i|W_j) = \\ &-\frac{a+b}{N_{DB}}\left[\frac{a}{a+b}\log\left(\frac{a}{a+b}\right) + \frac{b}{a+b}\log\left(\frac{b}{a+b}\right)\right] \\ &-\frac{N_{DB}-a-b}{N_{DB}}\left[\frac{N_L - a}{N_{DB}-a-b}\log\left(\frac{N_L - a}{N_{DB}-a-b}\right)\right. \\ &\left.+\frac{N_{DB}-N_L-b}{N_{DB}-a-b}\log\left(\frac{N_{DB}-N_L-b}{N_{DB}-a-b}\right)\right] \end{aligned} \quad (5)$$

The significance of this equation is that the information gain of a visual word is captured by a simple function of the values a and b as shown in Figure 4.

Note that for a given location, we only need to compute this conditional entropy for visual words which actually occur in the images at that location. In theory, it is possible that there may exist some visual word which occurs at every location *except* one, in which case this visual word which does not occur at the given location is nevertheless very informative about that location. In practice, we assume no such features exist, which is supported by the observation that each visual word generally occurs in some small fraction of the images. Thus, for visual words not present at some location the conditional entropy $H(L_i|W_j) \approx H(L_i)$ and the information gain $I(L_i|W_j) \approx H(L_i) - H(L_i) \approx 0$, meaning that for any location there is negligible information gain associated with visual words which do not appear there.

Since the above definition of information gain depends upon having already clustered the data into visual words, we bootstrap the process by constructing a number of vocabulary trees for relatively small subsets of the data. We define information with respect to these smaller subsets, select the most informative features from each image, and finally construct a tree using only these informative features.

5. Voting Scheme

To find the best-matching database image for a given query image, we match each feature in the query image to a number of features in the database using a vocabulary tree. We use a simple voting scheme in which matched features from the database vote for the images from which they originate. To achieve better performance, we normalize the vote tallies by N_i (the number of features in a given database image i) and NN_k (the number of near neighbors returned for a given query feature f_k). In addition, we average the tallies over a local neighborhood of N_L images. Thus, the number of votes C_d for a database image d can be computed by looping over every feature in each image in a local neighborhood, and comparing it against each of the N_q features in the query image, producing the following triple-summation:

$$C_d = \frac{1}{N_L} \sum_{i=d-\frac{N_L}{2}}^{d+\frac{N_L}{2}} \frac{1}{N_i} \sum_{j=1}^{N_i} \sum_{k=1}^{N_q} \delta_{match}(f_j, f_k) \frac{1}{NN_k}$$

where $\delta_{match}(f_j, f_k) = 1$ when database feature f_j and query feature f_k are both quantized to the same visual word in the vocabulary tree, and $\delta_{match}(f_j, f_k) = 0$ otherwise. The number of near neighbors returned for each query feature f_k can be similarly computed as:

$$NN_k = \sum_{i=1}^{N_{DB}} \sum_{j=1}^{N_i} \delta_{match}(f_j, f_k) \tag{6}$$

In practice, we do not explicitly perform these computations for every image, instead using the vocabulary tree to efficiently compute the sums in time linear in the number of features N_q in the query image.

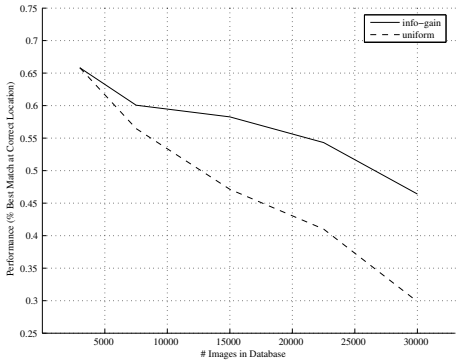


Figure 5. By building a tree using only the most informative features in each database image, we are able to improve performance over a tree built with training data uniformly sampled from the database. In all figures, *performance* is defined as the percentage of query images for which the top match in the database is within 10 meters of the ground truth location of the image.

6. Results

We have a database of 10 million images automatically acquired by driving a vehicle through a large city. Each image measures 1024x768 pixels and has an associated GPS coordinate (latitude and longitude), and a compass heading. In these experiments we use a 30,000 image subset corresponding to 20 kilometers of streetside data as depicted in Figure 1. We evaluate the performance of our location recognition method using a set of 278 query images acquired more than one year after the image database. All query images were captured with a 3.2 mega-pixel handheld consumer digital camera and labeled by hand with the latitude and longitude of the capture location. In these experiments, *performance* is defined as the percentage of query images for which the top match in the database is within 10 meters of the ground truth location of the image.

We perform two separate experiments – one to evaluate the effectiveness of using informative features to build vocabulary trees and the other to evaluate the performance of the Greedy N-Best Paths algorithm for vocabulary trees of varying branching factor. In the first experiment, we use a one million word vocabulary tree with branching factor $k = 10$ and depth $L = 6$ and the tree is searched using GNP with $N = 50$ paths. We build two different types of trees – one is built by hierarchically clustering features sampled uniformly from the database images, while the other is built by hierarchically clustering only the most informative features in the database. In both cases, a constant 7.5 million points are used in building the tree as we gradually increase the size of the database from 3000 images to 30,000 images, training new trees for each size of database. The entire 3000

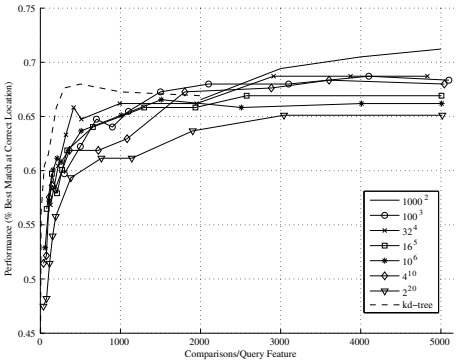


Figure 6. Vocabulary trees of any depth and branching factor can achieve comparable performance using GNP search. While vocabulary trees perform below comparable kd-trees using traditional $N = 1$ search, they can match and even exceed the performance-per-computation of kd-trees by using GNP search with increasing values of N. For each query feature, the number of comparisons against nodes in a vocabulary tree is equal to $k + kN(L - 1)$, which varies with N the number of best paths searched using the GNP algorithm. For the kd-tree, the number of comparisons per feature is the number of bins considered in the BBF algorithm.

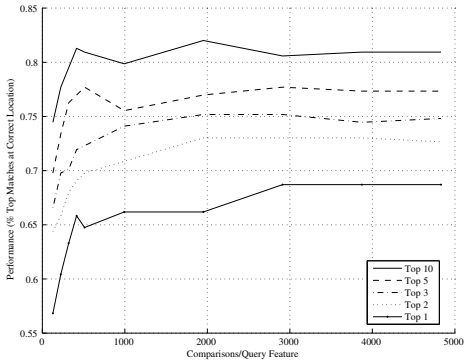


Figure 7. As we consider more than just the best matching result for the each of the 278 query images, the percentage of results within 10 meters of the ground truth location increases above 80%. These results show the performance of a 32^4 vocabulary tree on a 3000 image database using GNP with varying N.

image database consists of only 7.5 million features, and so both trees are identical at the beginning. As illustrated in Figure 5, as images are added to the database, the performance drops for both types of trees. However, the tree built from informative features degrades much more slowly than in the uniformly sampled case.

This result is significant because it suggests that the performance of a vocabulary tree is largely dependent upon its structure. Note that in both trees, the contents of the database are exactly the same, so that the only difference between the two trees is in the way they organize these database features into visual words. This result tells us that we would not want to use the same generic vocabulary tree to perform location recognition in two distinct cities, but rather that we should train such trees separately. This runs counter to the conclusion reached in [9] that generic vocabulary trees can be used as long as their leaf-nodes are filled with new data for each situation.

In the second experiment, we built trees of varying branching factor and depth as in Figure 2, all with approximately one million leaf nodes. We consider a database of 3000 images and compare the performance of each tree against the set of 278 query images using the GNP search algorithm with varying values for N, the number of paths searched. As discussed in Section 3.1, we can see from Figure 6 that performance varies with the number of nodes searched more strongly than with the branching factor of the tree. Though query time varies directly with the controllable number of nodes visited in the search, these experiments show that we can achieve close to maximum performance in only 0.2 seconds per query (excluding feature extraction) using a 32^4 tree and searching with 4-best paths.

In addition, in Figure 6 we compare the performance of these vocabulary trees against traditional kd-tree search. Because kd-trees using the Best-Bin First algorithm access the descriptors of every searched database feature, while vocabulary trees throw away this information, it is tempting to think that a kd-tree will naturally perform better at location recognition tasks and that one must compromise performance when using vocabulary trees. Just as GNP can be used to vary the amount of computation in a vocabulary tree search, the BBF algorithm can consider varying numbers of database features. If we equate the number of bins searched in the kd-tree with the number of nodes considered in the vocabulary tree, then we can directly compare the performance-per-computation of the two types of trees. While it is true that vocabulary trees using traditional $N = 1$ search perform below kd-trees that perform the same amount of computation, Figure 6 shows that vocabulary trees can match and even exceed the performance-per-computation of kd-trees by using GNP search with increasing values of N.

Finally, with respect to the overall performance results, note that in the time between the collection of the database and query images, many elements of the city had changed – businesses had hung up new signs and residences had been re-modeled, among many other changes, both major and minor. In addition, much of the query data was taken in poor lighting, leading to blurry and low-contrast images which more accurately reflect the quality of images one might expect from an average person unsure of their location. Under these circumstances, the greater than 70% recognition rate we achieved exceeded our expectations. Figure 7 also suggests that by employing a geometric consistency check on the top 10 matches, we could achieve performance of more than 80%.

Figure 8. Example database image sequences from commercial (top), residential (middle), and green (bottom) areas of a city. The significant overlap between consecutive images allows us to determine which features are most informative about each location.

Figure 9. Typical examples of the 278 query images (left) and the corresponding top matches returned from the database (right) using a 1000^2 vocabulary tree with $N = 4$.

7. Conclusion

In addition to demonstrating a system for large-scale location recognition, we have shown two new results with respect to vocabulary trees. First, we have found that the performance of a vocabulary tree on recognition tasks can be significantly affected by the specific vocabulary chosen. In particular, using the features that are most informative about specific locations to build the vocabulary tree can greatly improve performance results as the database increases in size. Second, we have shown that one can improve the performance of a given vocabulary tree by controlling the number of nodes considered during search, rather than by increasing the branching factor of the vocabulary tree.

References

[1] J. Beis and D. Lowe. Shape indexing using approximate nearest-neighbor search in highdimensional spaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

[2] Sergey Brin. Near neighbor search in large metric spaces. In *Int. Conf. on Very Large Data Bases*, pages 574–584, 1995.

[3] Keinosuke Fukunaga and Patrenahalli M. Narendra. A branch and bound algorithms for computing k-nearest neighbors. *IEEE Trans. Computers*, 24(7):750–753, 1975.

[4] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Journal of Theoretical Computer Science*, 38(2-3):293–306, June 1985.

[5] S. Lazebnik and M. Raginsky. Learning nearest-neighbor quantizers from labeled data by information loss minimization. In *International Conference on Artificial Intelligence and Statistics*, 2007.

[6] F. Li and J. Kosecka. Probabilistic location recognition using reduced feature set. In *IEEE International Conference on Robotics and Automation*, 2006.

[7] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[8] Andrew W. Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *Conf. on Uncertainty in Artificial Intelligence*, pages 397–405, 2000.

[9] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[10] Duncan Robertson and Roberto Cipolla. An image-based system for urban navigation. In *BMVC*, 2004.

[11] Hao Shao, Tomás Svoboda, Tinne Tuytelaars, and Luc J. Van Gool. Hpat indexing for fast object/scene recognition based on local appearance. In *CIVR*, pages 71–80, 2003.

[12] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477, 2003.

[13] M. Vidal-Naquet and S. Ullman. Object recognition with informative features and linear classification. *ICCV*, 2003.

[14] W. Zhang and J. Kosecka. Image based localization in urban environments. In *International Symposium on 3D Data Processing, Visualization and Transmission*, 2006.