

Adaptive Vocabulary Forests for Dynamic Indexing and Category Learning

Tom Yeh, John Lee, and Trevor Darrell
CSAIL, MIT
Cambridge, MA, USA
{tomyeh, jjl, trevor}@mit.edu

Abstract

Histogram pyramid representations computed from a vocabulary tree of visual words have proven valuable for a range of image indexing and recognition tasks; however, they have only used a single, fixed partition of feature space. We present a new efficient algorithm to incrementally compute set-of-trees (forest) vocabulary representations, and show they improve recognition and indexing performance in methods which use histogram pyramids. Our algorithm incrementally adapts a vocabulary forest with an inverted filesystem at the leaf nodes and automatically keeps existing histogram pyramid database entries up-to-date in a forward filesystem. It is possible not only to apply vocabulary tree indexing algorithms directly, but also to compute pyramid match kernel values efficiently. On dynamic recognition tasks where categories or objects under consideration may change over time, we show that adaptive vocabularies offer significant performance advantages when compared to a single, fixed vocabulary.

1. Introduction

It is rare that static representations are optimal in a dynamic world, yet most existing image indexing and category recognition models use a single fixed vocabulary representation. So-called "bag-of-words" models typically apply a batch clustering technique to a subsample of training data and perform vector quantization before storing any image records or computing category models. Hierarchical vocabulary representations such as the vocabulary tree [9] and histogram pyramid [3] have recently proven valuable for a range of image indexing and recognition tasks, but also use a single, fixed vocabulary model.

Many real world problems are dynamic in nature: the set of images and captions in news stories changes with current events; the set of objects to be recognized by a home robot depends on the particular home it inhabits; personal media to be organized visually will depend on the recent experiences of the user. While it is well accepted that model

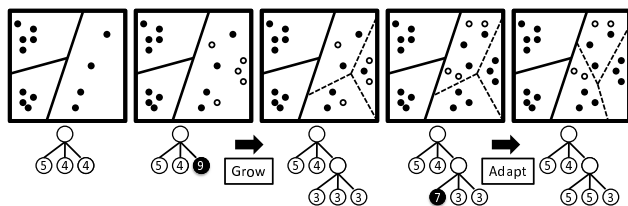


Figure 1. An adaptive vocabulary tree grows and adjusts partitions as new feature points (white dots) are continuously added.

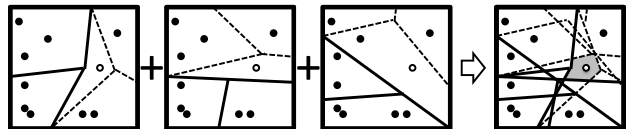


Figure 2. An ensemble of vocabulary trees can provide more regular partitions of the feature space.

parameters should evolve over time as categories change dynamically, it is generally presumed that a single, fixed visual word vocabulary representation—if large enough—will suffice for all tasks. We argue that this is not the case, and that performance suffers when representations are ill-tuned to the test distribution.

We propose an incremental vocabulary tree model that grows and adapts as new images are added to the database, based on a measure that encourages splitting nodes in the tree that become too ambiguous and pruning nodes that capture distinctions that are not relevant to the current set of tasks the system is facing (Figure 1). The computational cost per frame of our adaptive algorithm is minimal, and the resulting trees that are computed are similar in structure and performance to those computed offline.

Hierarchical clustering algorithms can lead to irregular partitions, so we adopt a set-of-trees (forest) approach with an ensemble of quantization grids (Figure 2). Randomized forest classification models have been recently shown to have theoretical and practical advantages in learning and vision tasks, and to improve performance with voting-based [5] or SVM-based methods [8]. We develop here an incremental method for computing a set-of-trees vocabulary representation which is applicable to those based on L1-

norm over histograms (e.g., vocabulary-tree [9] and video-google [12] methods), and to methods which compute a value based on histogram intersection across multiple histogram resolutions (e.g., the pyramid match kernel [4]).

Moreover, our method includes a new algorithm for efficiently maintaining pyramid match representations using an inverted filesystem storage model. Using our method a set of pyramid match representations (or kernel values) can be computed by simply traversing the maintained pyramids. We maintain an inverted filesystem (vocabulary tree), and can quickly update a forward filesystem (pyramid match) model.

We believe ours is the first method to provide representations suitable for both pyramid match and vocabulary tree computation, the first to incorporate set-of-tree models in these methods, and the first to incrementally compute vocabulary models for visual recognition. Incremental computation allows for adaptive models which outperform conventional static models in dynamic environments: on dynamic category recognition and image indexing tasks where individual categories or objects can be added or removed from an active set over time, we show that adaptive vocabularies offer significant performance advantages when compared to a single, fixed vocabulary.

In the following section we review related work on vocabulary models for visual recognition, and then describe the foundations of vocabulary tree and pyramid match representations relevant to our method. We then develop a vocabulary forest model for these hierarchical vocabulary models, and an incremental update algorithm which can keep current both an inverted (vocabulary tree) and forward filesystem (pyramid match) representation. We present experimental results and conclude with discussion of our method’s merits and suggestions for future work.

2. Related Work

Visual category recognition is an area of considerable interest in computer vision, and recognition methods based on local feature models [6, 7] have been demonstrated to have success across a range of tasks. However, search complexity with local feature models can be costly. To speed comparison over large databases, histograms over feature vocabularies were introduced in [12, 3] further demonstrated that a feature pyramid could approximate a correspondence-based distance measure. Efficient computation of vocabulary models was addressed by [9], who showed that hierarchical k -means could quickly compute very large vocabularies. There is a close relationship between the vocabulary tree of [9] and the feature pyramid of [3], although they are used for different computations. Both, however, have been heretofore limited to a single, static grid partitioning feature space. In this paper we develop an adaptive vocabulary tree model that is applicable to both approaches.

Forest models for classification were introduced to the vision literature by [5], who showed random tree models yielded highly competitive classification rates. Such models were recently used to directly define vocabulary modes for classification [8], however they were limited to SVM classification with a flat vocabulary model. In this paper we explore the use of forest models for histogram pyramid based models [9, 4], and further develop an efficient incremental scheme suitable for dynamic environments.

In the information retrieval community, the need for incremental clustering methods for dynamic environments has long been identified [13, 1, 15, 10]. Given the extremely high rate of update required by today’s online information system, it is important to pursue an incremental approach to updating clusters without having to perform complete re-clustering. Several deterministic and randomized incremental clustering algorithms have been proposed in [1] with the goal to minimize the number of clusters while enforcing a bound on the cluster diameter. However, their methods are not designed specifically for hierarchical clustering. In [10], an incremental algorithm has been developed to maintain bottom-up binary decision trees for hand-written digit recognition. A more sophisticated bottom-up hierarchical clustering method that allows varying branch factor can be found in [15]. This method first builds a “clustering feature tree” through a single scan of the data incrementally, and then applies global clustering to produce a small number of clusters. Finally, in [13], an incremental clustering method has been described for the task of classifying Web document into a “document cluster tree” to support efficient lookup. The method updates the tree dynamically by splitting over-crowded leaf nodes. Despite serving the same goal—indexing—a document cluster tree is similar to a vocabulary tree only in that both represent documents (or images) as a “bag-of-words.” However, they differ in that the former is binary and stores each document as a whole at a leaf node, whereas the latter is multi-branch and keeps each image as words separately across several leaves.

These incremental methods, though effective for their respective target tasks, are not readily applicable for a vocabulary tree for three reasons: (1) a vocabulary tree is constructed top-down by hierarchical k -means, (2) a vocabulary tree is not a binary decision tree, and (3) a vocabulary tree represents images at leaf nodes per-feature based on an inverted filesystem. Therefore, in this paper, we develop an incremental method specifically for vocabulary trees.

3. Background

Our method extends the well-known vocabulary tree [9] and vocabulary-guided pyramid match [3] methods for image indexing and category learning. We first review the aspects of these techniques that are integral to our method.

3.1. Vocabulary Tree

Given a large number of features extracted from training images as the input, we can learn a vocabulary tree using hierarchical k -means [9]. There are two parameters controlling the learning process: the number of levels L specifies the height of the resulting tree, and the branch factor B specifies the number of children each node has.

Each visual word is associated with an inverted file to record a list of images where the visual word can be found. Each entry in the inverted file is a pair $[i, c]$ where i is the index of an image and c is the count that specifies how many times the visual word appears in the image.

A vocabulary tree allows us to efficiently convert an input image I into a bag-of-words representation. We start by extracting features from the input image I . Then for each feature p in I , we want to find a corresponding visual word in the vocabulary tree. We begin the search from the root node of the tree. At each node, we visit every child node and compare p to each child node’s center (also represented as a feature point). We move to the child node whose center is the closest to p and continues the search, until we eventually reach a leaf node. The leaf node will be the visual word that corresponds to p .

3.2. Pyramid Match

The vocabulary-guided pyramid match, introduced by [3], is a fast method of approximating the minimum-cost partial match between two sets of high-dimensional vectors. It makes use of a hierarchical vocabulary to define the placement and sizes of the histogram bins. The similarity score between two pyramids is computed by matching points in a bottom-up fashion, weighting new matches at each level of the pyramid inversely proportionally to the bin size.

To generate a pyramid from a vocabulary tree and a new image, the points from the image are first embedded into the tree; each point traces a path down the vocabulary tree. For each such path, we create a parallel path in the pyramid. This way, the pyramid maintains a sparse representation, and similarity computations between two pyramids can be performed with one pass through each pyramid.

The vocabulary-guided pyramid match is useful because it allows us to match sets of high-dimensional vectors in linear time. When the weights are chosen to be inversely proportional to the size of the bin, we are guaranteed to have a Mercer kernel [3]. In Section 4.3 we describe how to efficiently maintain a set of pyramids corresponding to each of the images seen so far embedded in the current vocabulary.

4. Adaptive Vocabulary Forests

We consider recognition problems in a dynamic environment where new images are continuously added to the database and the recognition decision must be made for

each added image without delay. As the size of the database increases, the vocabulary tree used by a recognition system for indexing should also grow to allow the system to adapt to what images it actually encounters in the environment. We require causal recognition, so it is not possible to wait for all the images to be seen before making decisions.

Therefore, we have developed an adaptation method that incrementally grows a vocabulary tree as it sees more data, prunes obsolete nodes, maintains pyramid representations, and combines multiple trees into a forest to boost the recognition performance.

4.1. Growing Vocabulary Trees

Our tree growth method works in three steps:

1. A new point is inserted into the vocabulary tree
2. Overcrowded leaf nodes (if any) get removed in preparation for restructuring
3. The points displaced by the previous step are-clustered and inserted back into the vocabulary tree.

The behavior at each step is controlled by three parameters: the *capacity constraint* C which defines the maximum size of a leaf node, the *neighborhood size* S which sets the maximum number of points involved in any re-clustering step, and the *reduction factor* R which controls the maximum number of points allowed in a leaf node after a re-clustering step. In Section 5.3, we show that although these parameters provide varying time-space tradeoffs, the recognition performance remains fairly insensitive to them.

In step 1, we insert a new point from an image into the tree by placing it into the node whose center is closer to the point than any of the node’s siblings. We update the sizes of the nodes (if necessary) by recording the largest distance to the cluster center from any of the points in the node. In addition, we update the relevant pyramids to reflect the insertion of the new point (see Algorithm 1). If, as a result of the insertion, the number of points at the leaf w exceeds C , we trigger step 2 of our algorithm, which begins a process of *releasing* points in the neighborhood of the overcrowded node into a temporary pool.

In step 2, we release a number of nearby points no greater than the neighborhood size S (Algorithm 2). We take a greedy approach that starts by removing the visual word w that triggered the update, and continues by removing w ’s siblings one by one (from the closest to the furthest from w). Whenever a visual word is removed by this process, its points are also released into the pool. If there are no siblings left, and the pool of released points has not yet reached size S , the algorithm moves to the previous generation and begins removing w ’s parent and its siblings. During this removal process, however, we ignore unreleasable nodes

Algorithm 1 Insertion of a new point p from image i

```

procedure INSERT( $p, i$ )
   $w \leftarrow p$ 's matching visual word
  Add  $p$  to the inverted file at  $w$ 
  INSERTINTOPYRAMID( $p, i$ th pyramid)
  if  $|w| > C$  then
     $\Phi, x \leftarrow \text{RELEASEPOINTS}(w)$ 
    RE-CLUSTER( $\Phi, x$ )
     $\mathcal{P}_\Phi \leftarrow$  set of all pyramids with points in  $\Phi$ 
    UPDATEPYRAMIDS( $\Phi, \mathcal{P}_\Phi, x$ )
  end if
end procedure

```

whose descendants contain too large a number of points¹ to avoid pushing the size of the pool past size S . The remove-and-release process continues until all neighboring removable nodes have been removed.

In step 3, we begin the re-clustering process on the released points Φ . Let x be the lowest common ancestor of all removed nodes (i.e., the node at which the removal process stopped). Suppose x has m children remaining after the removal process. We run k -means clustering on Φ with $k = B - m$ to keep the branch factor constant throughout the tree. The reduction factor R is a ratio which specifies the constraint on the resulting distribution: if, as a result of the clustering, any of the nodes have more than $R * C$ children, we recursively perform more levels of k -means clustering rooted at that child until the constraint is satisfied.

4.2. Pruning Obsolete Nodes

As new feature points are continuously inserted into a vocabulary tree, at any given time period, some of the tree nodes may take on points while others remain unused. A prolonged period of inactivity at a node can indicate that the node is obsolete with regard to the current focus of the problem in a dynamic environment. To remove obsolete nodes, we maintain a list of access records to remember the least-recently used node: each time a node is accessed (i.e., a point is added to the a node), the node is moved to the front of the list. We specify a limit on the number of leaf nodes a tree can have. If the tree ever grows beyond the limit, we prune the least recently used leaf nodes until the limit is met. In Section 5.4, we show a pruned tree can improve performance on dynamic recognition problems.

¹Let Q be the minimum number of total points held by a node's descendants for the node to be considered unremovable. Then all removable candidate nodes must have at most $Q - 1$ points underneath them. Consider the node x at the highest level reached by the removal algorithm (i.e., the lowest common ancestor of all removed nodes). Because every node can have at most B children, x has at most B siblings which were also removed. Thus, the maximum number of released points must be $B(Q - 1)$. If we set $Q = \frac{S}{B} + 1$, we can ensure that we do not release more than S points.

Algorithm 2 Release from a visual word w 's neighborhood at most S points for re-clustering

```

function RELEASEPOINTS( $w$ )
   $x \leftarrow w$   $\triangleright$  maintain  $x$  to be root of all released nodes
   $\Phi \leftarrow x$ 's points
  while  $|\Phi| \leq S$  and  $x$  is removable do
     $\mathcal{Y} \leftarrow x \cup \{x$ 's siblings sorted by distance to  $x\}$ 
    for all  $y \in \mathcal{Y}$  such that  $|y| < \frac{S}{B} + 1$  do
       $\Phi \leftarrow \Phi \cup y$ 's points
      remove( $y$ )
    end for
    if  $x$  is removable and  $x$  has a parent then
       $x \leftarrow x$ 's parent
    end if
  end while
  return ( $\Phi, x$ )
end function

```

4.3. Pyramid Maintenance

In this section, we describe how our method keeps the pyramid representations of images up-to-date, stored in a forward filesystem. Note that maintenance of this representation is not necessary for the vocabulary tree to function normally, but is rather an optional component which, when included, can instantly return any of the maintained pyramids corresponding to the current vocabulary. Since every bin in each pyramid corresponds to a node in the vocabulary tree, we maintain at each node a list of pointers to the bins in the pyramids they correspond to. Note that this list is sparse; not every bin will contain a point from every image. When we retrieve the pyramid for a particular image, the bins report their sizes by following the pointer back to the vocabulary tree, which keeps the sizes up-to-date.

The two events during the incremental update of the vocabulary that trigger a change in the forward filesystem representations are when a (1) new point is embedded into the vocabulary tree, or (2) any points in the vocabulary tree are released and restructured.

To insert a new point, we first embed it into the vocabulary tree and follow a parallel path through the corresponding pyramid, creating new bins along the way if necessary. If a new bin is created, we immediately attach pointers from the corresponding nodes in the tree to the new bins.

If a newly inserted point causes a node in the vocabulary tree to exceed its capacity, thereby forcing local subtree to restructure itself, all of the affected pyramids must be modified accordingly. Let x be the root of the subtree containing all of the restructured nodes, and let Φ be the set of all points that were released (i.e., the union of all of the inverted files at the released leaves). From Φ we can determine the set of all affected pyramids (observe that although using the pointers stored in x for this task may seem more efficient,

Algorithm 3 Given a set of points Φ released from the vocabulary tree, the set of affected pyramids \mathcal{P}_Φ , and the root x of the restructured subtree, update the image pyramids

procedure UPDATEPYRAMIDS($\Phi, \mathcal{P}_\Phi, x$)

for all $\Delta \in \mathcal{P}_\Phi$ **do**

 REMOVEAFFECTEDBINS(Δ, x)

end for

for all $p \in \Phi$ **do**

$\Delta_p \leftarrow$ pyramid of p 's source image

 INSERTINTOPYRAMID(Δ_p, p)

end for

end procedure

procedure REMOVEAFFECTEDBINS(Δ, x)

$x_\Delta \leftarrow$ the bin in Δ pointed to by x

for all $c_\Delta \in \text{children}(x_\Delta)$ **do**

$c \leftarrow$ the node in the vocabulary pointed to by c_Δ

if c was removed by RELEASEPOINTS **then**

 remove(c_Δ)

end if

end for

end procedure

it would cause us to include more pyramids than necessary, since not all children of x were necessarily released and restructured). Given the set of affected pyramids, and pointers to the relevant subtrees in each of the pyramids (provided by x), performing the update is straightforward: we first delete all of the nodes corresponding to those that were released, and then insert each point in Φ into its respective pyramid, creating and linking new bins as necessary (as above). This is performed quickly by beginning the insertion procedure from x rather than from the root of the tree.

4.4. Combining Multiple Trees

There are two main benefits to combining the results of multiple incrementally-constructed vocabulary trees. First, a forest can reduce quantization effects near the boundaries of nodes in individual trees. Second, we can improve the performance during the early stages of exploration of a new area of the feature space (e.g., a new class of images is introduced). The incremental nature of our method makes it practical to use a set of trees because each tree is updated quickly; approaches that perform batch clustering become prohibitively costly.

For indexing, our goal is to identify the best overall candidate images. Given a query image, each vocabulary tree in the forest recommends a list of candidate images and their matching scores. We merge these lists into a single list. For each candidate on the merged list, we compute the total score by summing the scores from all lists. We rank the candidates according to their total scores and return them as

the retrieval result.

For category learning, we compute a single kernel based on a vocabulary forest. We simply define the pyramid match forest kernel to be the average of pyramid match kernels for each tree in the forest.

5. Experiments and Results

To evaluate and validate our approach, we designed five sets of experiments to address the following questions:

1. Can we combine several vocabulary trees into a forest to improve recognition performance?
2. Does the performance of a vocabulary tree suffer if it remains static rather than adapting to new data?
3. How do growth parameters affect the recognition performance, training time, and the size of the tree?
4. How does pruning a tree to limit its size affect recognition performance?
5. How does the performance of an incrementally grown tree compare to one trained using all of the data?

We answer these question in the context image indexing and category learning. For image indexing, we used the Zubud dataset [11], the UKY dataset [9], and a new FamousLandmark dataset. The Zubud dataset contains 1,005 images of city buildings in groups of five. The UKY dataset consists of 6,376 images in groups of four, ranging from daily objects to city scenes. The FamousLandmark dataset, which we created by collecting images from a popular online album service (Flickr), consists of 1,000 images of famous landmarks around the world (e.g., Empire State Building, Taj Mahal) in groups of ten.

We wish to analyze the performance of our algorithm in the context of dynamic real-world problems, such as online news stories, where images are added to the database very frequently and people tend to be interested in finding them. Such scenarios highlight the need to adapt quickly to new data so that recognition accuracy remains high for the current set of images. Our method tunes the vocabulary model based on the most recent inputs. Thus, we are most interested in the performance of our algorithm on test examples that reflect the current trend of data, and we compare our results to methods that do not adapt the vocabulary.

For indexing, we used the same image representation (128-d SIFT features extracted from maximally stable extremal regions) and scoring methods as [9]. To simulate a dynamic environment, at each time step i , we added one image X_i to the database in random order. Then we queried the database with X_{i+1} and retrieved matching images among the i images already indexed by the system. A retrieval attempt was considered *accurate* when the top-ranked retrieved image (1-NN) belonged to the same group as X_{i+1} (recall that the datasets consist of groups of matching images). After recording the accuracy, X_{i+1} was in-

serted into the system and X_{i+2} became the next query image. Note that at a given time, there may not be any image from a particular group in the database yet; a retrieval attempt could fail simply because the query image is the first instance of the group seen by the system.

We define *recent accuracy* at time i to be $A_T(i) = \frac{1}{T} \sum_{j=i-T}^i R(j)$ where T is the number of recent retrieval attempts under consideration, and $R(j)$ is a binary value that indicates whether the retrieval attempt on the j th image was accurate. By considering only retrieval attempts in the recent history, this metric can better reflect the current performance of the system. In our experiments, we set $T = 100$.

For category learning, we tested on the Caltech101 dataset [2], which contains 101 classes of object images. We used a one-vs-all SVM classifier and measured the performance using the *mean recognition rate* as in [4]: the score for a single class is the percentage of test examples of that class which were correctly classified, and the mean recognition rate is the average score over all classes. Since we closely followed the learning paradigm proposed by [4] based on pyramid match kernels, we also adopted an image representation similar to theirs (10-d PCA-SIFT features concatenated with 2-d coordinates, uniformly sampled over 8-pixel intervals). For the tasks reported below, SVM training was relatively fast due to the size of the input and was repeated every time the vocabulary was adapted. This method could become prohibitively costly in larger domains, and for those we would employ local SVM recognition methods [14].

5.1. Forests vs. Trees

The objective of the first set of experiments was to determine whether a vocabulary forest can improve the recognition performance of individual vocabulary trees. We trained five vocabulary trees using our adaptive method and tested the performance of a single tree versus a forest.

Figure 3 shows the benefits of forests for indexing (the first three graphs) and for category learning (fourth column). In our indexing experiments, larger forests always resulted in better instantaneous retrieval accuracy regardless how many images the system had already indexed and which dataset we tested on.

Similar results were found in object category learning. As the size of the forest increases (along the x -axis), the gain in accuracy (compared to the performance of a single tree) also increases. Here we report the results over all 101 classes in the dataset but with varying growth parameter settings. Figure 3 shows that a forest confers the greatest benefit to shallow trees composed of fewer words; this is useful to our method because the relevant branches will be shallow during early stages of expansion into new parts of the feature space (e.g., when new categories are first encountered).

5.2. Adaptive vs. Static Trees

In the second set of experiments, we studied the impact of adaptation (or lack thereof) on a vocabulary tree’s performance in a dynamic environment. For the indexing task, we simulated the dynamic nature of the problem by adding images to a vocabulary tree one at a time, and for the category learning task, images were added two classes at a time. After seeing varying amounts of data, we stopped adapting the trees and observed the effects on performance. We report the results of the experiments averaged over five runs.

Figure 4 shows that for indexing and for category learning, as more images are added to the database (along the x -axis), the performance of the vocabulary tree declines as the vocabulary stops adapting (i.e., where the performance curve branches off from the highest one). These results suggest that vocabulary trees in a dynamic environment should not stop growing; the moment adaptation stops, the ability to discriminate new images begins to deteriorate.

5.3. Impact of Growth Parameters

In this section, we describe the effect of the growth parameters C , S , and R introduced in Section 4.1 on the recognition rate, the number of times the tree is restructured, and the size of the tree. Each experiment was done for two different branch factors ($B = 6, 10$), and we always chose S to be proportional to C so as to keep the number of nodes affected by a re-clustering approximately constant.

Figure 5 shows that choosing a larger value of C causes both the size of the tree and the number of times we restructure it to decrease dramatically. Although we would normally expect the recognition rate to decrease as the number of nodes in the tree decreases, we attribute the stability in accuracy to the fact that S was chosen proportional to C , so that although restructuring happened less frequently for higher C , it was performed on more points. This shows that although more visual words in the vocabulary may lead to better performance simply due to more granularity in the feature space, the structure of the tree is also an important factor. This further highlights the need to adaptively update vocabularies: as the types of points being inserted into the tree change, it may be necessary to restructure parts of the tree to ensure a more useful distribution.

The next set of experiments tested the effect of S . We see that as the size of the neighborhood increases, fewer updates to the tree are necessary with a constant C . This is because after each re-clustering operation, the points in each of those nodes become better distributed (based to R , which we have held constant). Thus, as expected, size of the tree increases as S is increased. Observe that the recognition rate was not sensitive to the choice of S . This is because S has two competing effects on the structure of the tree: a higher value means that each re-clustering operation in-

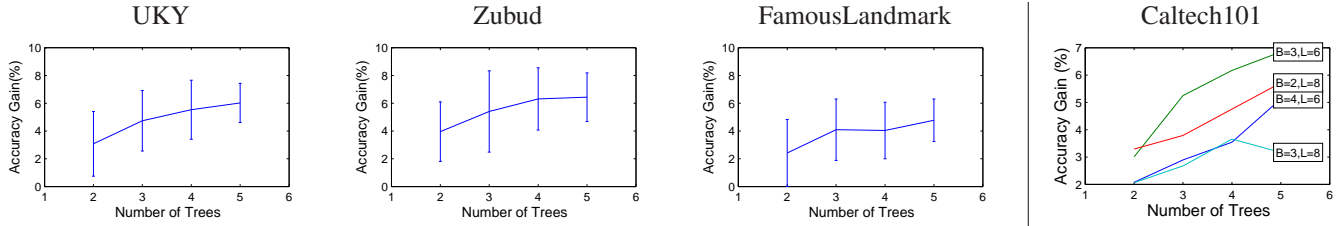


Figure 3. **Forests vs. Trees:** The accuracy gain (the absolute increase in accuracy over that of one tree) is greater when more trees are included in the forest. In indexing experiments (first three columns), results are averaged over five simulated runs of dynamic tree creation as images are inserted incrementally. In category learning experiments (last column), varying tree growth parameters were tested and plotted as separate curves (See Section 5.1).

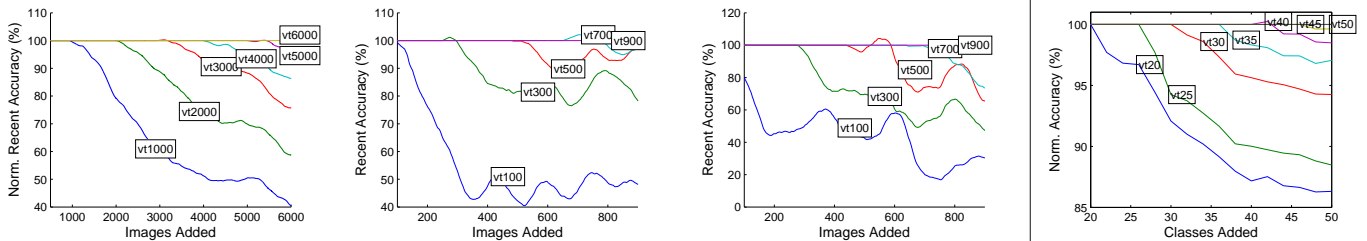


Figure 4. **Adaptive vs. static trees:** The performance of a vocabulary tree begins to decline when it stops adapting to new data. The label vtN refers to the performance curve of a vocabulary tree that stopped adapting after seeing N images in indexing experiments or N classes in category learning experiments. The curve for a particular vocabulary tree is normalized to the accuracy of that tree at the moment it stopped adapting (see Section 5.2).

cludes more nodes, causing larger substructures of the tree to be updated; at the same time, because the capacity constraint is triggered less often due to better overall point distribution after each re-clustering, the restructuring happens less often. Again, this suggests that the number of visual words present in the tree does not play as significant a role as the structure of the tree itself.

Figure 5 shows, as expected, that increasing R has the opposite effect as increasing S : with a high R , the criterion for redistribution is weak, leaving the resulting nodes possibly still very close to their capacities. This, in turn, causes updates to happen more frequently because the capacity constraint is triggered more often. As a result, the number of leaves in the tree becomes smaller but each becomes more populated.

Because varying the parameters does not significantly impact the recognition rate, they should be chosen according to the particular usage scenario (i.e., whether speed or size is favored). We have shown that having a smaller representation is not detrimental to performance because the structure of the vocabulary tree is changed as needed. However, such a representation is achieved at the cost of a higher number of updates to the structure.

5.4. Pruning

To study the effect of pruning on the resulting vocabulary, we simulated a dynamic category recognition scenario on the Caltech-101 database. After the vocabulary tree was shown a set of images from a given class, the recognition

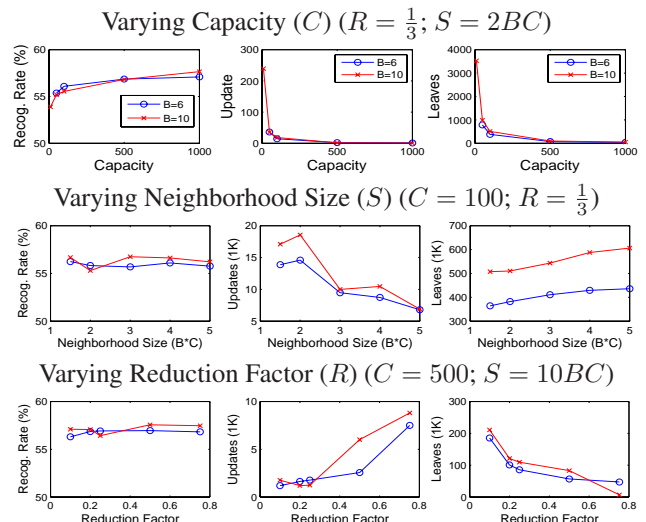


Figure 5. **Impact of growth parameters:** the recognition performance is insensitive to the parameters, but the size and speed can vary dramatically (see Section 5.3).

performance was tested on the 10 most recent classes it had previously seen, including the newly added one. We report the recognition performance averaged over 10 SVM runs with 15 training examples per class as well as the size of the tree in terms of the total number of leaves.

As expected, the total number of leaves in the tree (shown in Figure 6) remained constant for a pruned tree, while the number of leaves in the unpruned tree grew linearly as new images were added. We also observe that the

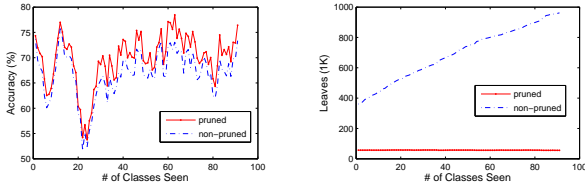


Figure 6. **Pruned vs. Non-pruned Trees:** Pruned trees achieve better accuracy but the sizes remain constant (see Section 5.4).

recognition rate for the most recently seen classes was always higher for the pruned tree than the unpruned tree, despite the limit on the number of visual words.

We believe that pruning is beneficial to the performance of an incrementally grown tree because it allows for the restructuring of heavily populated areas of the vocabulary tree. Recall that the restructuring algorithm chose to ignore nodes which contained too many points (“unreleasable” nodes) to avoid violating the neighborhood size constraint (S). As a result, structural improvements to those areas of the vocabulary tree cannot happen unless those nodes become obsolete and are pruned. Once pruned, that area can be repopulated with data from new categories, which may prefer a different distribution of the bins in that part of the feature space. As a result, the structure of the important parts of the vocabulary tree are kept up-to-date to reflect the most recently seen data. This finding further supports our claim that the structure of the tree is an important factor in determining the quality of the vocabulary; since different categories may prefer different distributions of the visual words in the same parts of the feature space, it is important to adapt the tree’s structure to newly seen images.

5.5. Incremental vs. Global Clustering

The objective of the last experiment was to demonstrate that trees incrementally grown would be comparable to the trees constructed from batch computation on the entire dataset. To this end, we created ten sets of problems with a varying number of Caltech101 classes. We applied both incremental and the baseline global strategies to train vocabulary trees and compare their recognition performances. Also, for simplicity, when forming the vocabulary, both incremental and baseline global methods have access to all the images in the given subset of images. We found that the differences in performance are small (avg. = $1.6 \pm 1.8\%$). Therefore, trees grown incrementally are comparable to trees obtained via batch computation, but are considerably faster and suitable for causal and dynamic recognition tasks.

6. Conclusion

We have presented an incremental method for learning a visual vocabulary in a dynamic environment where images are continuously added to the database. Empirically, we

demonstrated that: a forest of vocabulary trees outperform individual trees; the recognition performance declines once tree adaptation stops; with pruning, a vocabulary tree can be better attuned to the current problem; the setting of the growth parameters incurs time-space tradeoff but has small effects on recognition rates; and trees incrementally grown can achieve comparable recognition performance as trees learned from batch computation on the entire dataset.

References

- [1] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *STOC '97*, pages 626–635, New York, NY, USA, 1997. ACM Press. 3
- [2] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVPRW*, 12:178, 2004. 7
- [3] K. Grauman and T. Darrell. Approximate correspondences in high dimensions. In *NIPS '06*, 2006. 2, 3, 4
- [4] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. Technical report, MIT, 2006. 3, 7
- [5] V. Lepetit. Keypoint recognition using randomized trees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(9):1465–1479, 2006. Member-Pascal Fua. 2, 3
- [6] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004. 3
- [7] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004. 3
- [8] F. Moosmann, B. Triggs, and F. Jurie. Randomized clustering forests for building fast and discriminative visual vocabularies. In *NIPS '06*, 2006. 2, 3
- [9] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR '06*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society. 2, 3, 4, 6
- [10] A. Ribert, A. Ennaji, and Y. Lecourtier. An incremental hierarchical clustering. In *IAPR-VI'99*. 3
- [11] T. S. H. Shao and L. V. Gool. Zubud-zurich buildings database for image based recognition. Technical Report 260, Swiss Federal Institute of Technology, 2004. 6
- [12] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV '03*, page 1470, Washington, DC, 2003. IEEE Computer Society. 3
- [13] W. Wong and A. Fu. Incremental document clustering for web page classification. In *IS2000*, Japan, 2000. 3
- [14] H. Zhang, A. C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR '06*, pages 2126–2136, Washington, DC, USA, 2006. IEEE Computer Society. 7
- [15] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 25(2):103–114, 1996. 3