Fast Image Database Search using Tree-Structured VQ*†

Jau-Yuen Chen, Charles A. Bouman and Jan P. Allebach
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
{jauyuen,bouman,allebach}@ecn.purdue.edu

Abstract

In this paper, we exploit the techniques of tree structured vector quantization (TSVQ), branch and bound search, and the triangle inequality to speed the search of large image databases. Our method can reduce search computation required to locate images which best match a query image provided by a user. While exact search is possible, a free parameter allows search accuracy to be reduced, thereby providing a substantially better speed-up versus accuracy tradeoff.

1 Introduction

In recent years, there has been a growing interest in developing effective methods for searching large image databases based on image content. The interest in image search algorithms has grown out of the necessity of managing large image databases that are now commonly available on removable storage media and wide area networks. While manual browsing techniques are adequate for databases containing a few hundred images, a database of 100,000 images requires automated tools for search and perusal.

Example applications for image database search include graphic arts, or home photography. In these applications, characteristics such as color, texture and shape may be used to select a small number of images which are of potential interest.

Typical search algorithms require that the user provide a query image which has the desired characteristics. The search algorithm then returns the M best matches from the database of N images. Most search algorithms work by first extracting a feature vector for each image that contains the salient characteristics to be matched. Let q and x_i represent the K dimensional feature vectors for the query image and the i^{th} image of the database respectively. Let $S = \{i : 1 \leq i \leq N\}$ be the set of indices for all the images in the database. With this notation, the search process may be expressed as

$$i^* = \arg\min_{i \in S} d(q, x_i) \quad (1)$$

where $d(\cdot,\cdot)$ is a measure of the distance between feature vectors and i^* is the index of the best match. In this paper, we will assume that $d(\cdot,\cdot)$ is a metric so that it obeys the triangle inequality. A naive implementation of (1) requires a linear search of the entire database for the minimum distance image.

Interestingly, equation (1) appears in other applications such as vector quantization (VQ) and keyword search. In vector quantization, (1) corresponds to the encoding process when q is the source data and $\{x_i\}_{i \in S}$ are the set of N code words [1].

In both VQ and keyword search, it has long been known that the computation of (1) may be reduced by exploiting the structure of $d(\cdot,\cdot)$. In the VQ community, a variety of authors have exploited the triangle inequality to eliminate many matches that are not possible solutions [2] and similar strategies have also been applied in key word search [3, 4]. Recently, Barros, *et al* [5] and Berman and Shapiro [6] have applied the triangle inequality to improve search speed for image databases.

Perhaps the most widely studied method for speeding the solution of (1) is k-d trees [7]. In particular, k-d trees have attracted interest for application in fast image search [8, 9]. A k-d tree is a binary tree formed by comparing a single coordinate of the feature vector at each node of the tree. This constrains the tree to have decision hyperplanes that are orthogonal to a coordinate axis. This orthogonal binary structure can severely limit the optimality of the tree particularly when $K >> \log_2 N$, which is usually the case since, in practice, long feature vectors are important for achieving good matches. Most recently, White and Jain [10] and Kurniawati, Jin and Shepherd [11] have considered trees formed with less constrained structure.

In this paper we couple tree structured VQ (TSVQ) code books together with branch and bound search using the triangle inequality to create an efficient and flexible strategy for searching large image databases. An important

*This work was supported by Apple Computer.

†Appeared in the Proceedings of ICIP97, vol. 2, pp. 827-830, Santa Barbara, CA, Oct. 26-29, 1997.

advantage of our methods is that it allows computation and accuracy to be traded off through the choice of a single parameter λ. For $\lambda = 1$, the method yields the optimal match to a query, but only yields a moderate computational speedup. However, for $0 < \lambda < 1$ computation can be reduced by a factor of 20-40 while retaining good to excellent search accuracy on a database of 10,000 natural images. We test a variety of methods for designing the tree structured VQ and find that 10ary trees with mean centroids work best. In each case, the more general structure of the TSVQ appears to give a great advantage over more traditional k-d trees particularly when the dimensionality of the code words is high.

2 Tree Structured VQ (TSVQ)

Our method uses a TSVQ to represent the set of images in the database. Nodes of the tree are indexed by s. Each node is associated with a set of images $S_s \subset S$ and contains a code word z_s which is the centroid of S_s. The set $c(s)$ denotes the children of the node s which partition the images of the parent node.

$$S_s = \bigcup_{r \in c(s)} S_r$$

The leaf nodes of the tree correspond to the images in the database; so they are indexed by the set S. Each leaf node contains a single image, so for all $i \in S$, $z_i = x_i$ and $S_i = \{i\}$.

The tree is formed with a top down splitting strategy by using an algorithm similar to LBG or K-means. In each case, the distance norm $d(x_i, z_s)$ for clustering is an L_1 norm. For an L_1 norm, the centroid for LBG updates should be the median. However, we also consider modifications of LBG where the centroid is computed using the mean (L_2 centroid) or minimax code word.

3 Best First Search

Standard search methods for TSVQ's transverse directly from the root to a leaf of the tree choosing the branch which minimizes distance between the query, q, and the centroid, z_s, at each point. However, this search strategy is not optimum since it does not allow for back tracking.

Best first search is a well know method for searching trees [12] which keeps track of all nodes which have been searched, and always selects the minimum cost node to search further. While best first search is greedy, it does allow for backtracking. More formally, let $\underline{d}_s$ denote the cost or distance associated with a node s in the tree. Further assume that for any leaf nodes $i \in S$ $\underline{d}_i = d(q, x_i)$. (For the moment, we do not specify $\underline{d}_s$ for internal nodes of the tree.) Then best first search is defined by

$s^* = \text{root}$
$\Omega = \{s^*\}$
while s^* is not a leaf node
  $\Omega \leftarrow (\Omega - \{s^*\}) \bigcup c(s^*)$
  $s^* \leftarrow \arg\min_{s \in \Omega} \underline{d}_s$
end while

Notice that the minimization within the while loop may be efficiently implemented using a data structure known as a heap to keep track of the smallest current cost node. We will assume that the computational cost is dominated by the evaluation of the cost functions $\underline{d}_s$ and is therefore proportional to the number of nodes searched.

4 Branch and Bound Search

Under certain conditions best-first search is guaranteed to find the minimum cost solution. More specifically, if $\underline{d}_s$ is a lower bound on the cost of all leaf nodes that are its descendants, then this search is known as branch and bound or A* search [12] and the solution is guaranteed to be the global minimum. Then $\underline{d}_s$ must have the property that

$$\underline{d}_s \leq \min_{i \in S_s} d(q, x_i) \ .$$

A loose lower bound will result in a full search of the tree, but a tight lower bound will result in an efficient search which follows a direct path to the optimum node with little backtracking.

A useful lower bound results from the application of the triangle inequality to the distance metric $d(q, z_s)$. Define the radius of node s as

$$R_s = \max_{i \in S_s} d(x_i, z_s)$$

where z_s is the code word associated with node s and x_i is the feature vector of an image contained in leaf node i. Then for all $i \in S_s$

$$\begin{aligned} d(q, x_i) &\geq d(q, z_s) - d(x_i, z_s) \\ &\geq d(q, z_s) - R_s \end{aligned}$$

Using this inequality, we propose a general form for the cost function $\underline{d}_s$

$$\underline{d}_s = d(q, z_s) - \lambda R_s$$

where $0 \leq \lambda \leq 1$ is a constant.

When $\lambda = 1$ the search is guaranteed to yield the optimum solution. However, this bound is often too conservative. A smaller value of λ is often desirable since it can dramatically reduce the number of nodes to be searched while retaining good accuracy. If we reduce λ to 0, it will become a simple best first search based on the distance between query, q, and code word, z_s.

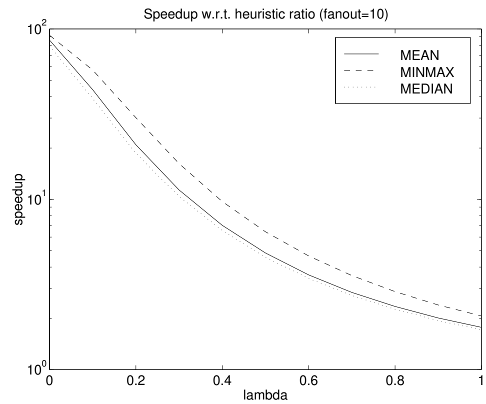

Figure 1: Speed-up versus the value of λ. Each plot is for a different centroid to the TSVQ clusters. The minimax centroid gives the best speed-up for exact search $\lambda = 1$.

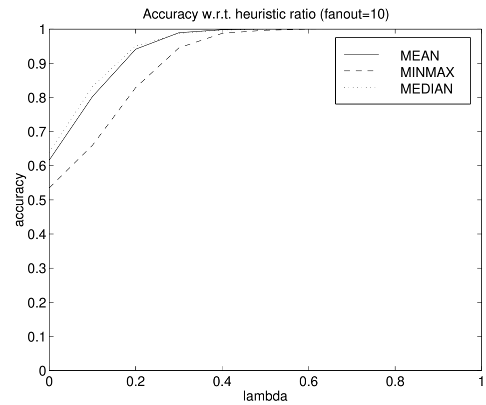

Figure 2: Search accuracy versus the value of λ. Each plot is for a different centroid to the TSVQ clusters. The minimax centroid gives uniformly poorer accuracy over a wide range of $\lambda = 1$.

When $\lambda < 1$ the accuracy can be improved by searching for more images than are required and then selecting the best from those that are returned. For example, if the user requests the 10 most similar images, these 10 images can be selected from the 20 best images found using the approximate branch and bound search. This strategy can result in a better accuracy/computation tradeoff.

5 Experimental Results

In this section, we present the results of searches on a database of 10,000 natural images. Our feature vector was of length 211 and was formed from histograms of color, texture, and edge information [13]. In all cases, the L_1 norm was used as the distance metric. A TSVQ was designed in a coarse-to-fine procedure as described in section 2. Unless otherwise stated, a fan-out of 10 was used (i.e. 10ary tree), and centroids were computed using feature means. For each example, 20 iterations of the LBG or modified LBG algorithm were performed using initial code words that were randomly selected from the corresponding group of image feature vectors.

Unless otherwise stated, we selected the 10 best image matches after searching for 20 images. (i.e. We searched for 10 extra images.) Each plot was then formed by averaging results over a set of 1000 randomly selected query images representing approximately 10% of the database. Computational speed-up is defined as

$$\text{speed-up} = \frac{\text{\# of images } (N)}{\text{\# of tree nodes searched}} \,.$$

We note that since the total number of tree nodes is $2N-1$, the speed-up can be less than one, but this was never observed.

λ **selection** - Figures 1 and 2 show the speed-up and accuracy as a function of the free parameter λ. The three plots illustrate the results using mean, median and minimax centroids. It is interesting to note that for $\lambda = 1$ (exact search) the minimax centroid gives better speed-up than the mean and median [11]. However, the speed-up for exact search is limited to approximatly 2 which is much less than can be achieved with approximate search.

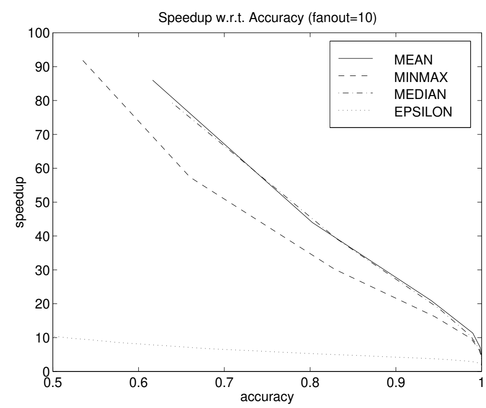

Figure 3: Speedup versus search accuracy. This plot shows that over a wide range of search accuracy the mean and median centroids are preferable to the minimax.

However, Fig. 3 shows the direct tradeoff of speed-up versus accuracy. In this plot it is clear that for even very high accuracy levels the mean and median centroids substantially out perform the minimax centroid. In practice, accuracies greater than 90 percent are probably not necessary for this application. This indicates that speed-ups of 20-40 are possible with good to excellent accuracy. Based on these results, the mean centroid seems preferable since it is much more efficient to compute than the median centroid. Figure 3 also show that the new search algorithm substantially outperforms the $(1+\varepsilon)$ approximate search

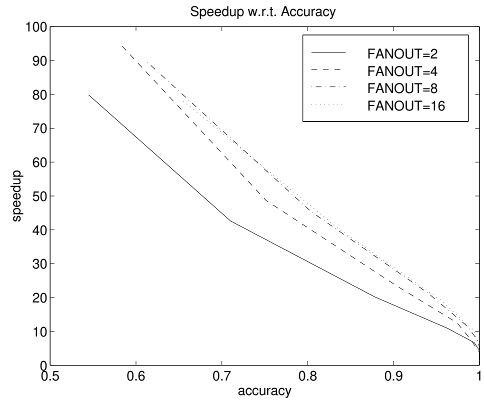

Figure 4: Speedup versus search accuracy for different tree fan-outs. Higher tree fan-outs yield better performance, but require more computation for tree construction. A fan-out of 8 appears sufficient.

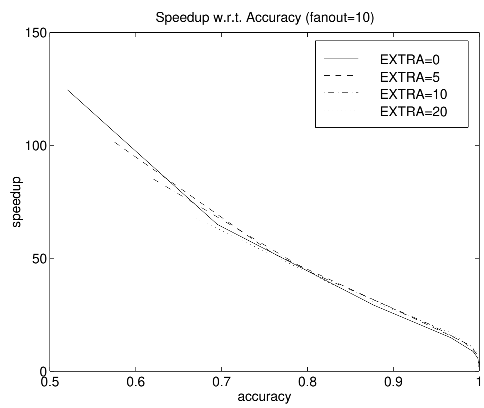

Figure 5: Speed-up versus search accuracy for different numbers of extra returned images. Returning extra images can improve the speed-up/accuracy tradeoff, but 50% extra returns appears sufficient.

proposed by Arya *et al* [14].

Fan-out - Figure 4 shows the search performance for a variety of different fan-out rates in the tree. A tree with low fan-out requires less computation to construct since the computation of the LBG algorithm is of order $N(\text{fan-out})(\text{tree depth})$. However, Fig. 4 indicates that a fan-out less than 8 can substantially reduce performance.

Extra returns - Figure 5 shows that searching for extra images can improve performance. However, the effect is rather weak. This is because returning extra images substantially increases the computation, thereby reducing the accuracy/speed-up tradeoff. Figure 5 indicates that returning 5 extra images is sufficent to gain most of the benefit.

6 Conclusion

In this paper, we show that branch and bound search based on the triangle inequality can dramatically speed up search on TSVQ structures. While exact search is possible, better overall performance is achieved by using approximate search. For approximate search, the best performance resulted from TSVQ structures with a fan-out>8, mean centoids, and searches which returned 50% extra images.

References

[1] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer Academic Publishers, 1991.

[2] M. T. Orchard, “A fast nearest-neighbor search algorithm,” *Proc. of IEEE Int'l Conf. on Acoust., Speech and Sig. Proc.*, May 1991, Toronto, Canada, pp. 2297–2300.

[3] W. A. Burkhard and R. M. Keller, “Some approaches to best-match file searching,” *Communications of the ACM*, vol. 16, no. 4, pp. 230–236, April 1973.

[4] V. Ruiz, “An algorithm for finding nearest neighbors in (approximately) constant time,” *Pattern Recognition Letters*, vol. 4, pp. 145–157, July 1986.

[5] J. Barros, J. French, W. Martin, P. Kelly, and M. Cannon, “Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval,” *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases IV*, vol. 2670, February 1-2 1996, San Jose, CA, pp. 392–403.

[6] A. Berman and L. Shapiro, “Efficient image retrieval with multiple distance measures,” *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases V*, vol. 3022, February 13-14 1997, San Jose, CA.

[7] R. F. Sproull, “Refinements to nearest-neighbor searching in k-dimensional trees,” *Algorithmica*, vol. 6, pp. 579–589, 1991.

[8] D. A. White and R. Jain, “Similarity indexing: Algorithms and performance,” *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases IV*, vol. 2670, February 1-2 1996, San Jose, CA, pp. 62–73.

[9] R. Ng and A. Sedighian, “Evaluating multi-dimensional indexing structure for images transformed by principal component analysis,” *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases V*, vol. 2670, February 1-2 1996, San Jose, CA, pp. 50–61.

[10] D. A. White and R. Jain, “Similarity indexing with the SS-tree,” *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases IV*, vol. 2670, February 1-2 1996, San Jose, CA, pp. 62–73.

[11] R. Kurniawati, J. S. Jin, and J. A. Shepherd, “The SS+-tree: An improved index structure for similarity searches in a high-dimensional feature space,” *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases V*, vol. 3022, February 13-14 1997, San Jose, CA, pp. 110–120.

[12] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

[13] J.-Y. Chen, C. A. Bouman, and J. P. Allebach, "Stochastic models for fast multiscale image search," *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases V*, vol. 3022, February 13-14 1997, San Jose, CA, pp. 133–144.

[14] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," *Proc. of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994, pp. 573–582.