Using Rival Penalized Competitive Clustering for Feature Indexing in Hong Kong's Textile and Fashion Image Database

Irwin King Lei Xu Laiwan Chan

Department of Computer Science & Engineering
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
{king,lxu,lwchan}@cse.cuhk.edu.hk

Abstract— **Efficient content-based information retrieval in image databases depends on good indexing structures of the extracted features. While indexing structures for text retrieval are well understood, efficient and robust indexing structures for image retrieval are still elusive. In this paper, we use the *Rival Penalized Competitive Learning* (RPCL) clustering algorithm to partition extracted feature vectors from images to produce an indexing structure for *Montage*, an image database developed for Hong Kong's textile, clothing, and fashion industry supporting content-based retrieval, e.g., by *color*, *texture*, *sketch*, and *shape*. RPCL is a stochastic heuristic clustering method which provides good cluster center approximation and is computationally efficient. Using synthetic data, we demonstrate the *recall* and *precision* performance of nearest-neighbor feature retrieval based on the indexing structure generated by RPCL.**

Keywords—**Rival Penalized Competitive Learning (RPCL), clustering, image database, indexing methods.**

I. INTRODUCTION

Montage is an image database developed for Hong Kong's textile, clothing, and fashion industry supporting content-based retrieval, e.g., by *color*, *texture*, *sketch*, and *shape* [3]. The system's search engine can be used in designing products, cataloging, image archiving, and image authentication applications. A Montage system catalog window with a fashion database is shown in Figure 1.

One of the key issues in information retrieval of images in Montage is the implementation of an efficient and robust indexing structure of the feature vectors. Without a properly designed indexing structure, the retrieval of information will be reduced to a linear exhaustive search. A good indexing structure, on the other hand, will make the information retrieval perceptually accurate and computationally efficient.

Problem Defined

Let $DB = \{I_i\}_{i=1}^{N}$ be a set of image objects. Without loss of generality, a feature extraction function $f : I \times \vec{\theta} \rightarrow \mathcal{R}^d$ extracts from an image I, with a set of parameters $\vec{\theta} = \{\theta_1, \theta_2, \cdots, \theta_m\}$, a real-valued d-dimensional vector. Hence, we may view the extracted feature vector as a point in a d-dimensional vector space. Furthermore, we may use a random variable X to denote the feature vector extracted from the image set DB and $\vec{x}_i, i = 1, 2, \cdots, N$ to denote the instance of the feature vector extracted from DB.

The set of feature vectors extracted from image data objects forms a sparse multidimensional feature space. It is natural to assume that there exists an underlying distribution of these feature vectors. More often than not, the underlying distribution of these feature vectors will not be uniform. This implies that with a suitable clustering scheme, the feature space can be partitioned naturally. With this partition, an indexing structure can be generated to make nearest-neighbor feature retrieval efficient.

A query to retrieve images with similar features in the image database is often given as $\vec{\hat{x}}$ which means to retrieve the set $\{\vec{x} \mid 0 \leq \mathcal{D}(\vec{x}, \vec{\hat{x}}) \leq \epsilon\}$ where ϵ is a tolerance bound for similarity match and $\mathcal{D}(\cdot, \cdot)$ is a distance function satisfying (1) $\mathcal{D}(\vec{x}, \vec{y}) \geq 0$, (2) $\mathcal{D}(\vec{x}, \vec{y}) = \mathcal{D}(\vec{y}, \vec{x})$, (3) $\mathcal{D}(\vec{x}, \vec{y}) = 0$ *iff* $\vec{y} = \vec{x}$, and (4) $\mathcal{D}(\vec{x}, \vec{z}) \leq \mathcal{D}(\vec{x}, \vec{y}) + \mathcal{D}(\vec{y}, \vec{z})$. In our case, we use the L_2-norm distance measurement defined as $L_2(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\| = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2}$. A nearest-neighbor search in the image database is a traversal of the indexing structure to a node which contains a cluster of similar feature vectors that satisfy the aforementioned criteria.

For nearest-neighbor retrieval, it is natural to group feature vectors together that are generally retrieved together in response to the requested query. This leads to clustering, a partitioning process producing mutually exclusive regions in the feature space in a way that is meaningful in the application domain context, of feature vectors which are extracted from images.

The problem is then to find an efficient way to generate a suitable clustering of the feature vectors so that an indexing structure can be produced for accessing data objects based on feature similarity.

Previous Work

There are many ways to produce clusters suitable for indexing in databases, e.g., R-tree [2], Quadtree [5], general hierarchical clustering methods, and VP-tree [7]. However, none of these methods is suitable for all situations. All seem to fail in the case when a requested nearest-neighbor query falls near the partition boundary since a similar feature vector near the query vector may be clustered in another partition. Although the R-tree, and VP-tree methods are adequate in many situations, they produce poor results since these methods partition the feature space without paying attention to the input distribution. On the other

0-7803-4859-1/98 $10.00©1998 IEEE

hand, although hierarchical clustering methods are more accurate, they are often computationally intensive resulting in impractical use for a large set of feature vectors.

In the next section, we outline the solution based on the RPCL method we have implemented to solve the feature space indexing problem. Experimental results are given in Section 3. We will then conclude with some discussion in the last section.

II. PROPOSED SOLUTION

There are two main goals in our proposed solution: (1) find a quick way to partition the input feature set into partitions and (2) impose an indexing structure over these partitions so that the nearest-neighbor information retrieval can be made effectively.

Rival Penalized Competitive Learning (RPCL) clustering [6] can be regarded as a kind of unsupervised extension of Kohonen's supervised learning vector quantization algorithm LVQ2 [4]. RPCL is a stochastic clustering algorithm that is able to perform adaptive clustering efficiently and quickly leading to an approximation of clusters that are statistically adequate.

The proposed solution is to use RPCL to find hierarchical clusters such that the indexing structure can be created based on a natural partition of the feature vector distribution. Although this may not result in a balanced tree structure, this indexing structure will be able to answer nearest-neighbor queries more effectively.

The main advantages of RPCL are: (1) the heuristic is computationally efficient and (2) RPCL can be implemented in a distributed environment achieving even greater speed-up in generating indexing structure of feature vectors.

A. Rival Penalized Competitive Learning Algorithm

Assuming there are k cluster centers, the basic idea behind RPCL is that in each iteration, the cluster center for the winner's unit is accentuated where as the weight for the second winner, or the rival, is attenuated. The remaining $k-2$ centers are unaffected. The winner is defined as the cluster center that is closest to the randomly selected feature vector. In our application, we use the special version of the RPCL clustering algorithm when $k=2$. In other words, we only have a winner and a rival (second winner).

Let k, $\vec{c}_w$ and $\vec{c}_r$ to denote the number of clusters, cluster center points for the winner and rival clusters respectively.

Step 0: **Initialization** Randomly pick $\vec{c}_1$ and $\vec{c}_2$ as the initial cluster centers.

Step 1: **Winner-Take-All Rule** Randomly take a feature vector $\vec{x}$ from the feature sample set X, and for $i=1$ and 2, we let

$$u_i = \begin{cases} 1, & \text{if } i=w \text{ such that} \\ & \gamma_w \|\vec{x}-\vec{c}_w\|^2 = \min_j \gamma_j \|\vec{x}-\vec{c}_j\|^2, \\ -1, & \text{if } i=r \text{ such that} \\ & \gamma_r \|\vec{x}-\vec{c}_r\|^2 = \min_j \gamma_j \|\vec{x}-\vec{c}_j\|^2, \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where w is the winner index, r is the second winner (rival) index, $t=1,2,\cdots,\gamma_j = n_j / \sum_{i=1}^{k} n_i$ and n_i is the cumulative number of the occurrences of $u_i = 1$. This term is added to ensure that every cluster center will eventually become the winner somehow. It is called the Frequency Sensitive Competitive Learning (FSCL) [1] as an algorithm that reduces the winning rate of the frequent winners.

Step 2: **Updating Cluster Centers** Update the cluster center vector $\vec{c}_i$ by

$$\Delta\vec{c}_i = \begin{cases} \alpha_w(\vec{x}-\vec{c}_i), & \text{if } u_i = 1, \\ -\alpha_r(\vec{x}-\vec{c}_i), & \text{if } u_i = -1, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

where $0 \le \alpha_w, \alpha_r \le 1$ are the learning rates for the winner and rival unit, respectively.

Step 1 and 2 are iterated until one of the following criterion is satisfied: (1) the iteration converges, (2) $\alpha_w \to \epsilon$ for a time decaying learning rate of α_w with a pre-specified threshold of ϵ, or (3) the number of iterations reaches a pre-specified value. In practice α_c and α_r often depend on time with $\alpha_c(t) \gg \alpha_r(t)$. One useful function is

$$\alpha_c(t) = \alpha_c^i \left(\frac{\alpha_c^i}{\alpha_c^f}\right)^{\frac{t}{t_{max}}}, \tag{3}$$

where α_c^i and α_c^f are the initial and the final learning rate and t_{max} is the maximum iteration step taken.

B. Generating Top-Down RPCL Indexing Structure

After the RPCL clustering is finished, the next step is to build an indexing tree with n sets of subtrees. Given a *bipartite* graph $G=(V,E)$, an indexing tree has the following properties: (1) there is one specially designated node called the *root* of the tree and (2) the remaining nodes (excluding the root) are partitioned into $m \ge 0$ disjoint sets $G_1,\cdots,G_m$, and each of these sets in turn is a tree. The trees $G_1,\cdots,G_m$ are called the subtrees of the root. Each subtree in the binary indexing structure will be a cluster partition. For example, at the root level, all feature vectors are in one cluster and at depth i would have 2^i subtrees (or clusters).

There are two ways of performing the top-down RPCL clustering. The first is to use the full feature space each time to cluster. The other method is a hierarchical approach which will cluster the next level of feature vectors based on the subset of data points partitioned in the previous level. In this report we used the full feature space approach to cluster the whole feature space into 2^i partitions for every i stage.

After clustering the feature space into $2^i, i=1,2,\cdots$ partitions, we may map the feature vectors in the feature space into the indexing tree structure using the inverted file technique. At the top level, a query $\vec{x}$ is compared to the two cluster centers, $\vec{c}_1$ and $\vec{c}_2$. The data objects in the cluster with the minimum L_2 measurement will be selected for the search to proceed. This traversal of the indexing structure will terminate when the number of elements in the node (cluster) reaches the desired goal.

III. EXPERIMENTAL RESULTS

We tested the RPCL clustering algorithm in generating the indexing structure with 6,400 2-dimensional synthetic generated feature vectors. Let $\vec{\mu} = (\mu_1, \mu_2, \cdots, \mu_n)$ and $\vec{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_n)$, the mixture of n Gaussian distributions is defined as $N(\vec{\mu}, \vec{\sigma}^2)$ with the generating function defined as $g(x) = 1/(\sigma\sqrt{2\pi})\exp[-[(x-\mu)^2/2\sigma^2]], -\infty < x < \infty$. For simplicity, we use a constant σ for all Gaussian mixtures, i.e., $\vec{\sigma} = 0.1$.

Given the set of *a priori* clusters, $\vec{C} = \{c\}_1^n$ and the calculated RPCL cluster, $\vec{C'} = \{c'\}_i^m$, The performance measures of RPCL is defined as follows:

$$\text{Recall} = \frac{\text{Number of target images retrieved}}{\text{Number of target images}} = \sum_{c_i \in \vec{C} \wedge c'_j \in \vec{C'}} \frac{c_i \cap c'_j}{\#c_i} \quad (4)$$

$$\text{Precision} = \frac{\text{Number of target images retrieved}}{\text{Number of image retrieved}} = \sum_{c_i \in \vec{C} \wedge c'_j \in \vec{C'}} \frac{c_i \cap c'_j}{\#c'_j} \quad (5)$$

where $\#c_i$ denotes the number of elements in the cluster c_i.

Using an Ultra Sparc I, we conducted 100 trials with the same 2-dimensional feature vectors but with different initial of RPCL cluster centers in order to find the average performance of Recall and Precision of Eqs. (4)-(5). The results are summarized in Tables I and II.

Integration into the Montage Image Database

Figure 2 shows the color sketch-pad query window and its results of fabric images currently available in Montage. In addition to the R-tree indexing in Montage, we are implementing RPCL indexing into the Montage system framework. The feature extraction function produces a feature vector which is composed of the primary indexed color of mutually exclusive partitioned image lattice. Our preliminary experiments showed similar recall and precision performance results as those found in the synthetic experiment.

IV. DISCUSSION

We found a problem when the number of RPCL clusters is greater than the actual number of Gaussian clusters in the input feature distribution. In this case, multiple RPCL centers can be bunched together spatially. This leads to an incorrect assessment of clusters. However, in practice the number of natural clusters in the input distribution is quite large so this problem only occurs in rare situations.

Rival Penalized Competitive Learning clustering is a stochastic heuristic clustering method which provides good approximation of cluster centers and is computationally efficient in generating indexing structure for nearest-neighbor information retrieval. Our experimental results have shown RPCL's *recall* and *precision* performance using synthetic feature vectors. We are now integrating RPCL into the Montage system for content-based retrieval of nearest-neighbor query.

ACKNOWLEDGEMENTS

This work was supported in part by the Hong Kong Research Grant Council, CUHK4176/97E and IDTC Grant, AF/17/95.

REFERENCES

[1] Stanley C. Ahalt, Ashok K. Krishnamurthy, and Prakoon Chen. Competitive learning algorithms for vector quantization. *Neural Networks*, 3(3):277–290, 1990.

[2] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD*, pages 47–57, 1984.

[3] I. King, A. Fu, L. W. Chan, and L. Xu. *Montage: An Image Database for the Hong Kong's Textile, Fashion, and Clothing Industry.* The Chinese University of Hong Kong, 1995. `http://www.cse.cuhk.edu.hk/~viplab`.

[4] T. Kohonen. The self-organizing map. *Proc. IEEE*, 78:1464–1480, 1990.

[5] H. Samet. The quadtree and related hiearchical data structures. *Computer Surveys*, 16(2), 1984.

[6] Lei Xu, Adam Krzyżak, and Erkki Oja. Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. *IEEE Trans. on Neural Networks*, 4(4):636–649, 1993.

[7] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.

TABLE I
RECALL TABLE.

Mixture Groups	*RPCL Clusters*				
	4	8	16	32	64
2	0.503	0.264	0.122	0.181	0.108
4	0.991	0.582	0.275	0.201	0.185
8	0.977	0.988	0.508	0.334	0.325
16	0.947	0.903	0.836	0.597	0.457

TABLE II
PRECISION TABLE.

Mixture Groups	*RPCL Clusters*				
	4	8	16	32	64
2	0.621	1.000	0.991	1.000	1.000
4	0.722	1.000	1.000	0.999	0.986
8	0.509	0.828	0.694	0.890	0.836
16	0.207	0.418	0.522	0.592	0.783

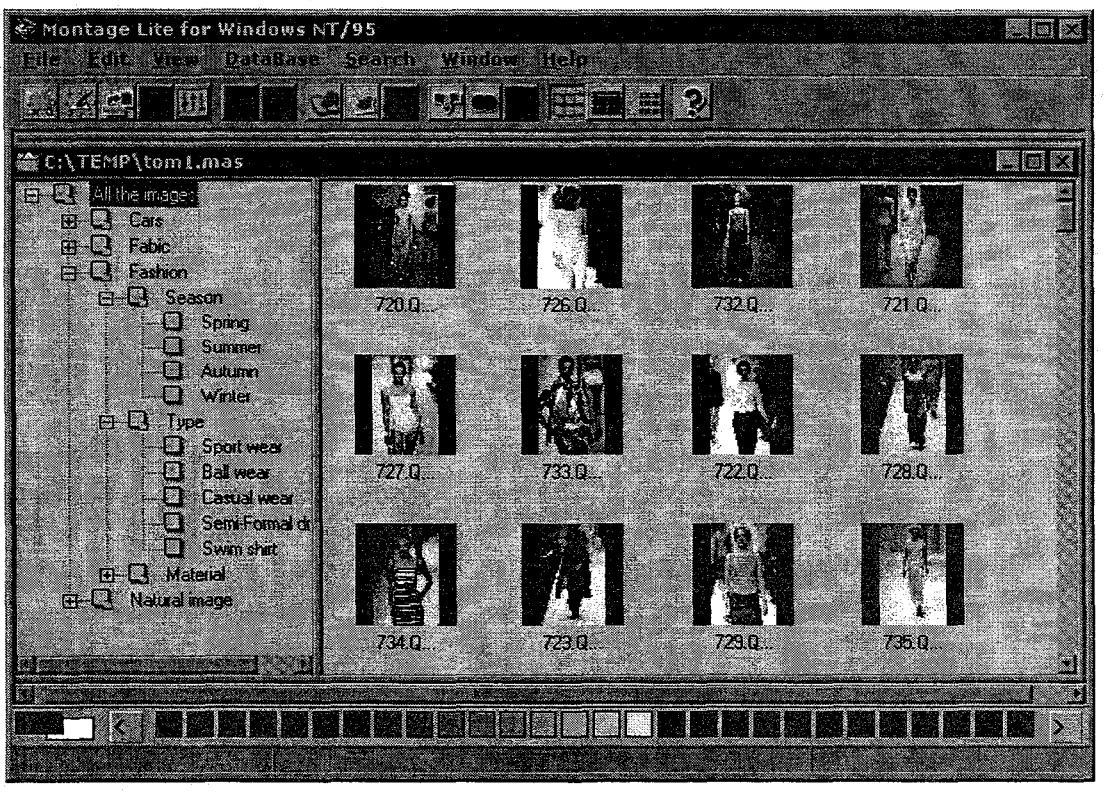

Fig. 1. The Montage image catalog system window (`http://www.cse.cuhk.edu.hk/~viplab`).

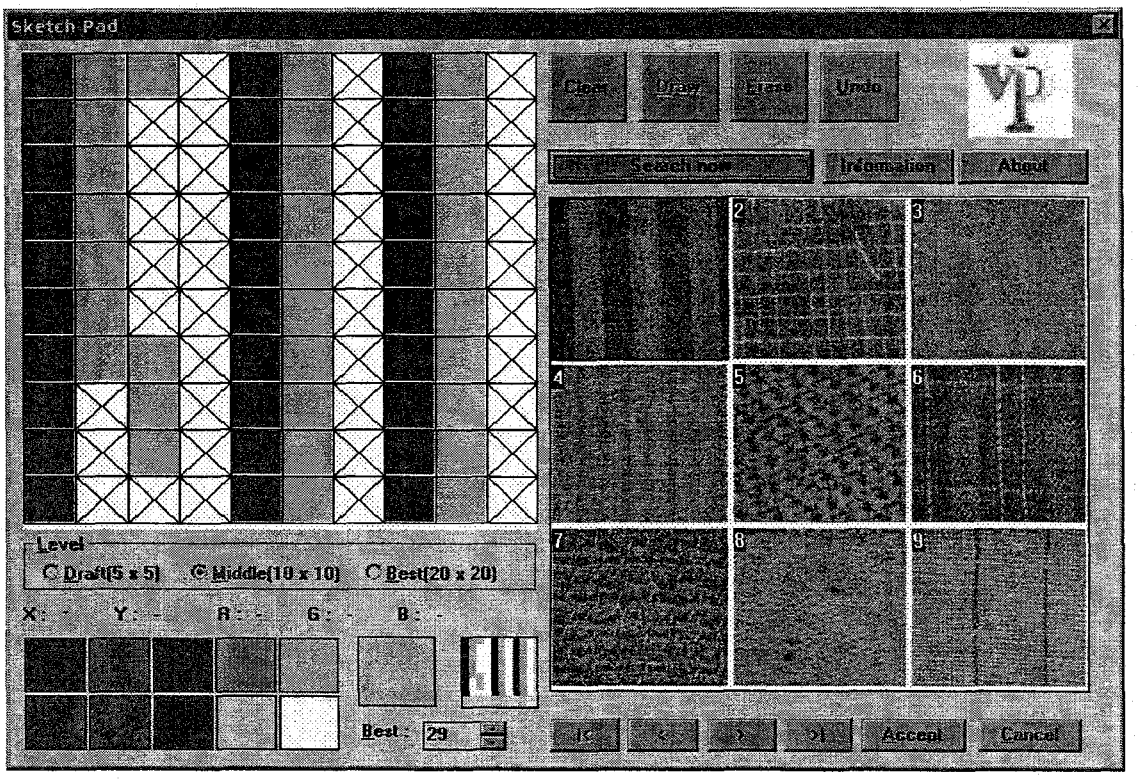

Fig. 2. A screen-shot of the color sketch-pad window with its query result of fabric images.