A Hybrid Data and Space Partitioning Technique for Similarity Queries on Bounded Clusters*

Piyush K. Bhunre[1], C.A. Murthy[2], Arijit Bishnu[3], Bhargab B. Bhattacharya[2], and Malay K. Kundu[2]

[1] National University of Singapore, 3 Science Drive 2, Singapore 117543
[2] Indian Statistical Institute, 203, B. T. Road, Kolkata - 700108, India
[3] Indian Institute of Technology, Kharagpur, Kharagpur - 721302, India

Abstract. In this paper, a new method for generating size-bounded clusters is proposed such that the cardinality of each cluster is less than or equal to a pre-specified value. First, set estimation techniques coupled with Rectangular Intersection Graphs are used to generate adaptive clusters. Then, the size-bounded clusters are obtained by using space partitioning techniques. The clusters can be indexed by a Kd-tree like structure for similarity queries. The proposed method is likely to find applications to Content Based Image Retrieval (CBIR).

1 Introduction

In a CBIR system, the images are indexed as points in the multidimensional feature space (mostly Euclidean space). Given a query image, the same features are extracted to perform a similarity query that basically is a search for the nearest neighbour(s) of the query point, or a range search around the query point in an Euclidean space [1]. Most of the nearest neighbor (NN) search techniques try to reduce the search time by using multidimensional indexing techniques. Multidimensional data structures can be classified as (i) *space partitioning* (e.g. Kd-tree and K-D-B tree) based techniques and (ii) *data partitioning* (e.g. R tree, R^+ tree, R^* tree, SS tree, and SR tree) based techniques [2]. The *space partitioning* techniques recursively partition the entire space into mutually disjoint sub-spaces. In *data partitioning* based index structures, the data points at a particular level are obtained by clustering them in the sibling nodes using different bounding regions like rectangular hyperbox, sphere, etc. For a survey on these, see [2]. Unlike the *data partitioning* index structure, range query on the multidimensional space using a *space partitioning* index structure like Kd-tree, requires only a single check at each node corresponding to the splitting dimension. Apparently these

* This work was supported in part by a grant from Intel Corp., USA (PO #CAC042717000).

S.K. Pal et al. (Eds.): PReMI 2005, LNCS 3776, pp. 544–550, 2005.

methods offer logarithmic search time, but may be worse than linear search if $d > \log n$[3]. For all the above techniques, the problem increases manifold when the secondary memory, that is inherently non-random access, comes into play owing to huge image databases. On the other hand, clustering techniques have also been in use for image retrieval. Generally, clustering algorithms do not have any control over the number of points in a cluster. In addition, indexing clusters of arbitrary shape and size remains a problem. For such clusters spread over many disk pages, search time will be very high.

To address the above issues, we propose a method that generates clusters of bounded size (the size is an input to our method), which then can be indexed using a two-level multidimensional data structure for retrieval. We use set estimation techniques based on the data distribution to form a bounding box around each data point and then use rectangular intersection graphs to find clusters. Next, we use space partitioning techniques to further sub-divide the clusters so that the size of each cluster is less than the bounding size specified. This space partitioning technique is also used for indexing these bounded-size clusters.

2 Set Estimation and Clustering

In set estimation problem [4], the parameter to be estimated is a set. The cluster $\alpha(\subset I\!R^d)$, which is unknown is to be estimated. Below is the definition of the set of parameter $\mathcal{A}$ and a probability measure P_α on $\alpha \in \mathcal{A}$.

Definition 1. *[5,6]* $\mathcal{A} = \{\alpha : \alpha \subseteq I\!R^d$, *$\alpha$ is path connected, compact,* $cl(int(\alpha)) = \alpha$ *and* $\partial\alpha$ *consists of finitely many analytic arcs* $\}$ *Here cl represents "closure", int represents "interior" and* $\partial\alpha$ *represents the boundary of* α *and is defined as* $\alpha \cap cl(\alpha^c)$.

Definition 2. *[6] The properties of* P_α *are as follows: (i)* $P_\alpha(\alpha^c) = 0$, *(ii)* $P_\alpha(\partial\alpha) = 0$; *(iii)* $P_\alpha(A \cap \alpha) > 0$, $\forall$ *open set* $A \subseteq I\!R^d$ *with* $A \cap \alpha \neq \phi$, *and (iv) there exists a function* $f : I\!R^d \rightarrow I\!R^+$ *such that* $P_\alpha(A) = \int_A f d\mu$, $\forall$ *subset A in* $I\!R^d$.

Based on the above definition of the probability measure, we have the following definition for a consistent estimate of α.

Definition 3. *Let* $X_1, X_2, \cdots, X_n$ *be a random sample from* P_α. *Then* α_n *obtained from* X_i*'s is said to be a consistent estimate of* α *if*

$$\lim_{n \to \infty} E_{P_\alpha}(\mu(\alpha_n \Delta \alpha)) = 0 \tag{1}$$

where, μ *is a Lebesgue measure in* $I\!R^d$ *and* Δ *denotes symmetric difference and E denotes expectation [4].*

2.1 Estimation of α

In set estimation, depending on the sample points, we estimate the path connected set from which the sample points have come. We can estimate α by α_n

in the following way as proposed in [4,6]. Let d denote the dimension of the data. Let $\delta^n = (\delta_{n1}, \delta_{n2}, \cdots, \delta_{nd})^t \in I\!R^d$ with $\delta_{nj} > 0, \forall j$ such that the following conditions are met:

Condition 1. *(i)* $\delta_{nj} \to 0$*; and (ii)* $n \prod_{j=1}^{j=d} \delta_{nj} \to \infty$ *as* $n \to \infty$.

Define $A_i = \{X \in I\!R^d : |x_{ij} - x_j| \leq \delta_{nj}, \forall j = 1, 2, \cdots, d\}, \forall i = 1, 2, \cdots, n$. Then, form α_n as a union of connected subsets as $\alpha_n = \bigcup_{i=1}^{i=n} A_i$. It may be noted that α_n is taken as the union of closed axis-parallel hyper-rectangular neighborhoods around each sample point X_i. We now need to choose a suitable neighborhood so that Condition 1 and the consistency condition in Equation 1 are satisfied.

2.2 Choice of the Neighborhood δ_{nj}

We select the δ^n-neighborhood of the sample points as follows: the ranges $R_j = Max\{x_{ij}\}_{i=1}^{i=n} - Min\{x_{ij}\}_{i=1}^{i=n}$ (where x_{ij} is the j^{th} component of X_i) for each of the components of the sample points are found out. Then, we choose each δ_{nj} as $\delta_{nj} = \frac{R_j}{n^{\frac{1}{2d}}}, \forall j = 1, 2, \ldots, d$.

Lemma 1. $\delta_{nj} = \frac{R_j}{n^{\frac{1}{2d}}}, \forall j = 1, 2, \ldots, d$, *satisfies Condition 1.*

Proof. $\lim_{n\to\infty} \delta_{nj} = \lim_{n\to\infty} \frac{R_j}{n^{\frac{1}{2d}}} = 0$. Also, $\lim_{n\to\infty} n \prod_{j=1}^{j=d} \delta_{nj} = \lim_{n\to\infty} \sqrt{n} \prod_{j=1}^{j=d} R_j = \infty$.

For a proof of the consistency property (see Equation 1 in Definition 3) of the above δ^n, see [6]. From the above discussions, we have the following observations.

Observation 1. *Any* $\alpha \in \mathcal{A}$ *is a path connected set by definition 1 and its estimate* $\alpha_n \to \alpha$ *as* $n \to \infty$*[6] with respect to the consistency condition (1).*

Observation 2. *The set of all samples, which belongs to a connected subset is path connected and forms an estimate of the cluster of the sample point set. Thus, finding an estimate of the clusters of the sample points is same as finding the connected component of* α_n.

2.3 Generation of Clusters Using Rectangular Intersection Graph

The estimated set α_n is a union of some connected subsets with respect to the neighborhoods of the sample points.

Observation 3. *To find a path connected set, we observe that there is a path between two data points if their corresponding hyperrectangles intersect and a cluster is formed by a maximal set of data points such that there is a path between each pair of vertices.*

From the above observation, we can find the path connected set as a cluster by finding the strongly connected component in a Rectangular Intersection Graph (RIG). The *RIG*, $G = \{V, E\}$ is formed as follows. For each data point X_i, we

assign a vertex $v_i \in V$ of the graph. Any two vertices v_i and v_j have an edge e_{ij} between them if the neighborhoods $N_{\delta^n}(X_i)$ and $N_{\delta^n}(X_j)$ intersect, i.e., $N_{\delta^n}(X_i) \cap N_{\delta^n}(X_j) \neq \phi$. Finding the connected component of an *RIG* having n vertices (i.e. n sample points) takes $O(n \log n)$ time and $O(n)$ space for $d = 2$ [7]. For $d > 2$, forming the *RIG* is equivalent to finding the k intersecting pairs in a set of n axis-parallel hyper-rectangles in d dimensions and can be done in $O(n\log^{d-1} n + k)$ time using only $O(n)$ space [8]. The *RIG* thus formed has $O(k)$ edges. We can find its connected component in $O(n+k)$ time [9]. Thus, the overall time needed for finding the cluster according to our method is dominated by the *RIG* formation and is $O(n\log^{d-1} n + k)$. Thus, from the above discussion, we have the following.

Theorem 1. *The clusters can be found as connected components in an RIG of the consistent estimate α_n of a path connected set α and can be computed in $O(n\log^{d-1} n + k)$ time.*

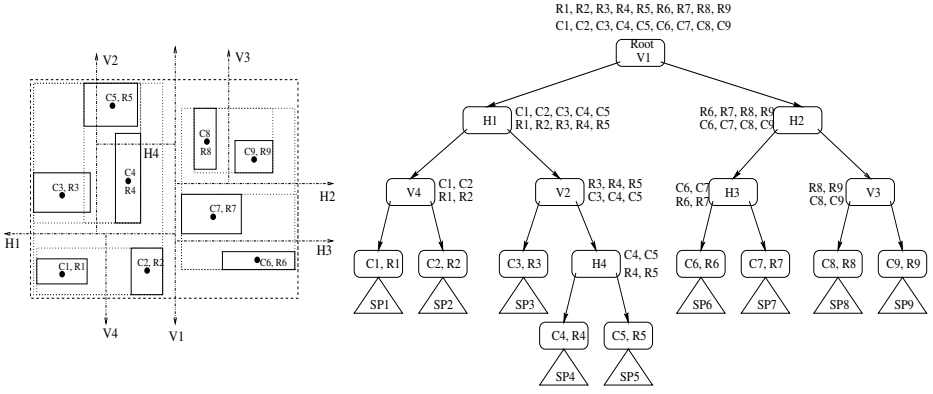
Remark 1. The clustering algorithm developed here is adaptive. A cluster of any shape (convex and non-convex) can be computed by the above method. The nearest neighbor of a point belongs to the same cluster.

3 Generation of Bounded Clusters and Their Indexing

We now have set estimated clusters, whose size and shape can be arbitrary. To partition the data and index it into small buckets, each of whose size does not exceed a threshold value (T), we use a two-level space partitioning tree.

We first find the set C of cluster centroids, $C = \{C_1, C_2, \ldots, C_m\}$, and the enclosing axis-parallel hyperboxes of each of the m clusters. These hyperboxes may be overlapping. We will now induce a recursive space partitioning, by splitting the centroids along a single dimension at each split, based on the spread of their enclosing hyperboxes. From the spread of the enclosing axis-parallel hyperboxes, we find the dimension j ($\leq d$), along which the data set has the largest elongation. Next, we find the median M_j of the j^{th} component of the cluster centroids. Then, we split C into two subsets $C^{(1)}$ and $C^{(2)}$ such that $C^{(1)}$ contains all cluster centroids whose j^{th} component is less than M_j. Form $C^{(2)}$ as $C^{(2)} = C \setminus C^{(2)}$. To ensure that the splitting plane does not pass through any cluster centroid, we fix the splitting discriminant co-ordinate M_{nj} as the average of the maximum and minimum j^{th} coordinates of $C^{(1)}$ and $C^{(2)}$ respectively. This technique is applied recursively until each subspace has only one cluster centroid left. Thus, we generate a Kd-tree [7] using this method. At each split, we store the discriminant value M_{nj} at the tree node.

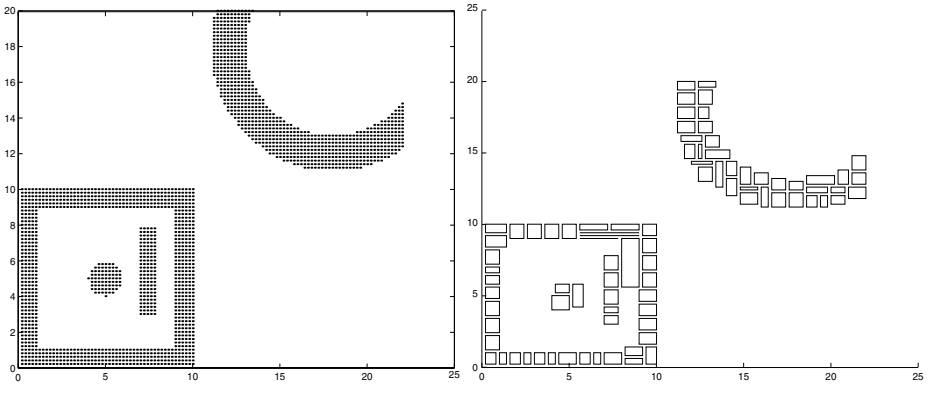
After the space partitioning for the cluster centroids ends, we have the first level Kd-tree, and corresponding to each cluster, we have the data points in that cluster. Now, we use the same recursive sub-division and stop when the cardinality of a set of points is just less than T, thus ensuring that the data is partitioned into buckets, each of which has size less than T. Note that, this has also a Kd-tree structure but unlike the level-1 tree, this level-2 tree may



(a) Rectangles containing CC's. C: cluster centroid, R: enclosing boxes, V: vertical cut lines, H: horizontal cut lines

(b) DSP-tree of CC's in (a). SP: the level-2 tree formed by the points in that cluster.

Fig. 1. An example of space partitioning and indexing structure in 2D



(a) Connected components of synthetically generated points in $I\!R^2$

(b) A program output showing splitting boxes generated by our method

Fig. 2. An example of adaptive clustering and space partitioning

not be balanced because of the arbitrary sized natural clusters generated using set estimation techniques. Henceforth, we term the two-level tree thus generated as a *DSP* tree (an acronym for Data and Space Partitioning). See Fig. 1 for an example. Fig. 2 shows an example of the adaptive cluster formation and its splitting into size-bounded clusters.

4 Similarity Query and Results

The image database used is the Columbia Object Image Library(COIL-20) [10] of 20 objects. There are 72 images per object, taken at pose intervals of 5 de-

grees, making a total of 1440 gray-scale images. We have taken feature vectors of dimension 7 whose first three components are the set of three invariant moments [12] and the other four components are the elements of the Euler vector [11], which is a 4-tuple, where each element is an integer representing the Euler number of the partial binary image formed by the gray code representation of the four most significant bit planes of the gray-tone image. As query, we have taken the front view of each of the 20 objects and an axis-parallel range query hyperbox around them. We have also performed the exhaustive search for testing the validity of the result of the experiment performed by our method. For each query, we have reported the first 10 (atmost) among the retrieved images. An output image is said to be accepted if it belongs to same COIL-object as the query image. The acceptance percentage is calculated as the ratio of the number of accepted output images by number of reported images. The results are given in Table-1. The purpose of the experiments reported here is not to achieve a high retrieval success for the COIL database, but to show that our scheme gives comparable retrieval success to exhaustive search with an appreciable savings in computation time.

Table 1. Search results

	DSP-tree Serach				*Exhaustive Search*			
Input Images	No. Of Ret. Im.	No. Of Acpt. Im.	% of Acpt.	Time (sec.) T_{DSP}	No. Of Ret. Im.	No. Of Acpt. Im	% Of Acpt.	Time (sec.) T_{EX}
obj1.raw	10	5	50	0.01	10	5	50	0.06
obj2.raw	10	6	60	0.00	10	6	60	0.06
obj3.raw	10	4	40	0.02	10	4	40	0.07
obj4.raw	10	10	100	0.00	10	10	100	0.05
obj5.raw	7	7	100	0.00	7	7	100	0.05
obj6.raw	10	1	10	0.02	10	1	10	0.05
obj7.raw	10	1	10	0.01	10	1	10	0.05
obj8.raw	10	6	60	0.00	10	6	60	0.05
obj9.raw	10	10	100	0.00	10	10	100	0.05
obj10.raw	10	4	40	0.01	10	4	40	0.06
obj11.raw	10	1	10	0.02	10	1	10	0.07
obj12.raw	10	2	20	0.01	10	2	20	0.07
obj13.raw	10	7	70	0.00	10	7	70	0.05
obj14.raw	10	4	40	0.01	10	4	40	0.05
obj15.raw	10	5	50	0.01	10	5	50	0.06
obj16.raw	10	5	50	0.02	10	5	50	0.06
obj17.raw	10	9	90	0.02	10	9	90	0.06
obj18.raw	10	8	80	0.01	10	8	80	0.05
obj19.raw	10	4	40	0.00	10	4	40	0.06
obj20.raw	10	2	20	0.00	10	2	20	0.06
Average	total = 197	total = 101	51.2690	0.0085	total = 197	total = 101	51.2690	0.0550

5 Discussions and Conclusion

In this paper, we first generated adaptive clusters using set estimation techniques and rectangular intersection graphs. Secondly, we devised a two-level space partitioned indexing scheme to generate indexed clusters of bounded size. We have evaluated our scheme against the exhaustive search method. In the future, we will elaborate the technique adopted to split and index a cluster whose enclosing box is completely contained in another. Further experiments are to be carried on image databases of larger size and with better representative features.

References

1. D. A. White, and R. Jain, "Similarity indexing with the SS-tree", *in Proc. of the* 12^{th} *Int. Conf. on Data Engineering*, New Orleans, USA, pp. 516-523, Feb. 1996.
2. V. Gaede and O. Günther, "Multidimensional access methods", *ACM Computing Surveys*, pp. 170-231, vol. 30, No. 2, June 1998.
3. J. M. Klinberg, "Two algorithms for nearest neighbour search in higher dimensions", in Proc. 29^{th} *ACM Symposium on Theory of Computing*, 1997.
4. U. Grenander, *Abstract Inference*, John Wiley & Sons, 1981.
5. H. L. Royden, *Real Analysis*, Prentice Hall of India, New Delhi, 2000.
6. C. A. Murthy, *On Consistent Estimate of Classes in* $\mathbb{R}^2$ *in the Context of Cluster Analysis*, Ph.D. Thesis, Indian Statistical Institute, Kolkata, India, 1988.
7. F. P. Preparata and M. I. Shamos, *Computational Geometry*, Springer Verlag, New York, 1985.
8. H. Edelsbrunner and M. H. Overmars, "Batched dynamic solutions to decomposable searching problems", *Journal of Algorithms*, vol. 6, pp. 515-542, 1985.
9. T. H. Cormen, C. E. Leiserson and R.L. Rivest, *Introduction to Algorithms*, Prentice Hall of India, 1998.
10. S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library: COIL-100", *Technical Report CUCS-006-96*, Department of Computer Science, Columbia University, February 1996.
11. A. Bishnu, B. B. Bhattacharya, M. K. Kundu, C. A. Murthy, and T. Acharya, "Euler vector for search and retrieval of gray-tone images", *IEEE Trans. SMC-B*, vol. 35, no. 4, pp. 801-812, 2005.
12. T. H. Reiss, "The revised fundamental theorem of moment invariants", *IEEE Trans. PAMI*, vol. 13, no. 8, pp. 830-834, 1991.