KPYR: AN EFFICIENT INDEXING METHOD

T. Urruty [§] *F. Belkouch* [*] *C. Djeraba* [§]

[§]LIFL-UMR CNRS 8022. Laboratoire d'Informatique Fondamentale de Lille
Université de Lille 1, France {djeraba, urruty}@lifl.fr

[*] STID Dpt. Statistique et Traitement Informatique des Données
Université de Lille 2, France belkouch@iuts-mailstid.univ-lille2.fr

ABSTRACT

Motivated by the needs for efficient indexing structures adapted to real applications in video database, we present a new indexing structure named Kpyr. In Kpyr, we use a clustering algorithm to partition the data space into sub-spaces on which we apply Pyramid Technique [1]. We thus reduce the search space concerned by a query and improve the performances. We show that our approach provides interesting and performing experimental results for both K-Nearest Neighbors and Window queries.

1. INTRODUCTION

In recent years, the size of digital video archives has enormously grown. This is particularly so for, news agencies, TV broadcasters, and advertising agencies running large digital archives of movies and video footage. With the size of archives growing, the problem of locating certain sequences becomes more and more challenging.

Our work forms a part of an exploratory project which aims to improve the standardized representations with a validation on voluminous databases of company films. Companies of audio-visual sector have an immeasurable amount of unexploited video that grows continuously. Yet, the value of that information depends on how easily we can manage, find, retrieve, access and filter it. One of the needs of the professional users is to be able to find existing video sequences in order to re-use them in the creation of new films. Each video sequence is represented as a data point in multidimensional data space.

Sequences are located using two types of similarity queries: Window Query (WQ), also called range query, and K-Nearest Neighbors Query (KNN). This search depends on the used data access method which in turn depends first on how the data are stored, indexed in the memory and then, on how search method is used in these indexing structures.

Many indexing structures have been proposed in the literature. The Pyramid Technique [1] is the first indexing technique that is not strongly affected by the *curse of dimensionality*. This approach has been a reference for recent techniques. In short, all the existing indexing methods are not general and effective enough to enable an index structure that 1) supports both type of queries and 2) does not strongly depends on the workload (data size, dimensionality, data distribution…).

We propose in this paper a powerful indexing method called Kpyr. This method associates a well known clustering technique to a multidimensional indexing structure. Clustering allows homogeneous data clusters which can be indexed separately using Pyramid Technique. As the data distribution is uniform in each cluster, we obtain efficient indexing structures: a B+ tree for each cluster. Clusters will be then used to divide the space into regions in order to establish some geometrical forms which will reduce the search space of a query and thus improve the performances. The problem of searching data is reduced to a geometrical intersection problem.

Kpyr performs well for both Window and KNN queries under different workloads. Experiments have shown that our approach 1) guarantees a good quality of accurate results in a relatively short time and 2) outperforms existing methods. The outline of the paper is given as follow: Related work is presented in section 2. Our approach is described in Section 3. Implementation issues and performances are detailed in Section 4. Section 5 concludes the paper and addresses future works.

2. RELATED WORK

The study of the multidimensional indexing structures started with the work of [6] which proposes R-Tree, a reference in the field. Other alternatives of R-Tree appeared thereafter as the X-tree [7]. These structures have shown their limits for high dimensions (the *curse of dimensionality* problem). Other indexing methods match the problem of dimensionality by proposing a transformation of the space with dimension N into an index of one dimension. Among them:

Pyramid Technique [1] is the first one adapted to Window Query (WQ). The basic idea is to partition the data space into pyramids which will be then sliced into hyper-plane.

- The D dimension space is divided into 2D pyramids whose top is the center of the multidimensional space and the bases are D-1 dimensional surfaces.
- The pyramids are labeled with numbers from 1 to 2D.

A point is indexed by P_{ih}, the sum of the number i of the pyramid to which it belongs and the height from that point to the top h. This P_{ih} value is the indexing key in a B+tree[8].

Pyramid Technique is the reference technique among other multidimensional indexing methods recently developed [2, 3, and 4].

IMinMax [2] is based on a parameter θ adaptive to data distribution and adopts a simple mapping function that is computationally inexpensive. This technique also adapted to WQ is more dynamic than Pyramid's. However, the choice of θ value is not trivial and the performances depend on it.

IDistance [3] partitions the data and selects a reference point for each partition. Processing is mainly based on distance calculations. This technique is adapted to KNN Queries and may not be usable to WQ.

P+Tree [4] can be used for both Window and KNN queries. This approach combines a space division technique, Bisecting K-means [5] and the Pyramid Technique. The performances of this method strongly depend on the results of the space division technique.

Discussion

Similarity based search using the Pyramid technique indexing is achieved by a WQ search in a B+ tree. Its performances do not deteriorate when the dimension increases. However Pyramid Technique has two drawbacks: the first one concerns the data distribution. When this one is not uniform, selecting the top of the pyramids is a difficult task, which influences on the quality of the indexing. The second drawback appears when querying the points near a d-1 dimension hyper plane. A simple query near pyramid bases can generate a useless access to a very great number of data which has no similarity and affects the performances considerably.

Both IDistance and IMinMax are adapted to only one type of query respectively KNN and WQ. P+Tree can be seen as an improvement of the Pyramid Technique. However, it is not sufficiently effective. It is obvious that dealing with the degradation of the performances due to the data distribution or the position of the query, the solution must reduce the area (space data) accessed by a query especially when it is located at the corner or at the edge of the pyramid. The trivial idea is to divide the space into subspaces and then apply the Pyramid Technique in each subspace.

The effectiveness of such a solution depends naturally on how the data are partitioned and how reference points are selected. P+tree divides the space into subspaces which are essentially hyper rectangles and then applies the Pyramid Technique on each one. Their space division method is based on Bisecting K-means algorithm [5], but it is reduced to a simple partitioning according to only one dimension. That is not a real clustering technique. Consequently the top of the obtained pyramids are not "good" reference points. Assuming that the real queries follow the same distribution as data, they will not be located around the top of the pyramids. So, P+tree is not sufficiently powerful.

3. KPYR: OUR APPROACH

We present first an overview of our approach and then we give a formal description of our algorithm.

Informal description

Kpyr Algorithm follows three steps:

1) For our data clustering, we use K-means algorithm [9] which is a known and widely used clustering technique. K homogenous clusters are built. 2) For each one, we transform the corresponding subspace into a hypercube unit so that the Pyramid Technique can be applied (Transformation_base function). 3) We obtain a B+ tree as an indexing structure for each cluster.

For a WQ, we must be able to determine the zone (we call it the region) where the research must be done and then the clusters which intersect with WQ. In order to do so, we apply a border calculation algorithm based on the distribution of the clusters obtained by K-means as shown in Figure 1.

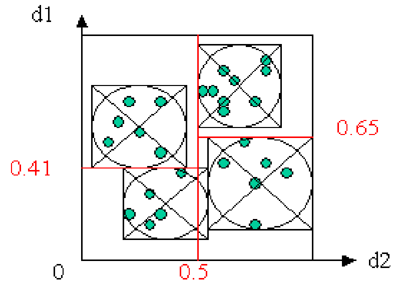

Figure 1. Geometrical representation of KPYR

Two search steps are necessary to perform a query: the first one uses a binary space tree to determine the clusters to analyze. The second one corresponds to the search of a WQ in the pyramids of the reached clusters. Our contribution is the preliminary use of a real classification method before the application of Pyramid Technique. The indexing is thus more efficient. Indeed, the space and the quantity of data needed to index by Pyramid method are smaller. The top of each pyramid is a real reference point for all points in the cluster. The computation of the border is solely done to determine the clusters concerned by the query which efficiently reduces the search space and therefore the response time.

P+tree also proceeds by a space division which does not always perform well. Let us take the simple case of a space with two dimensions. As shown in Figure 2a, such a data distribution generates a bad space division by P+tree which affects the performances. The application of our method provides a better division of space (Figure 2). It is obvious that this phenomenon becomes more critical in higher dimensions.

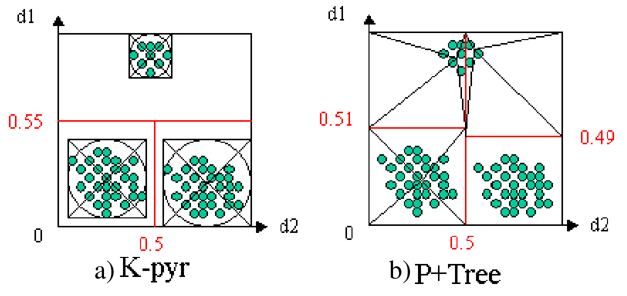

Figure 2a and 2b. Geometrical comparison

Pseudo code

We consider a data space of dimension d and size N. We denote the data points set RV: $RV=\{P_i \in R^d,\ 0<i<N\}$. This set will be clustered using K-means algorithm and provides K clusters C_i $(C_i \subseteq RV)$. The centre of a cluster C_i is denoted Cc_i $(Cc_i \in R^d)$. We use $Distance(P_i, P_j)$ to compute the distance between two points P_i and P_j in the space RV. Radius_i represents the radius of the cluster C_i $(Radius\_i = \{Max\_Distance(Cc_i,P_j),\ j \in C_i\}$. Hkmeans is a hash table storing all clusters with their centres. $(Hkmeans=Hashtable\{(Cc_i,C_i),0<i<K\})$.
Pyramid(S) is the pyramid algorithm applied to a set of data points S. It returns a B+Tree.

KPYR Indexing Algorithm

```
KpyrIndexing(Rv)
  // call the K-means algorithm
 Hkmeans  =  Kmeans(Rv)
 //apply Pyramid algorithm on each cluster
 For each cluster $C_i$ do
        $C_i'$= Transformation_base($C_i$)
        $B_i$ = Pyramid($C_i'$)
  End for
 //build the Binary Space Tree
 SpaceTree(C, SpaceTreeRoot)
End
```

Creation of the binary space tree

```
SpaceTree(Sc, SpaceTreeRoot)
//Stop if the sc contains one cluster
if |Sc|=1 go to End.
// determine the border of the region
For each dimension m do
         $V_m$ = ($Cc_{im}$ ordered), $0<i<K$
           $D_m$ = Max ($V_j+1-V_j$), $0<j<K$
           $V_m$ = $V_j$ / $D_m$ = Max ($V_j+1-V_j$)
End For
//determine the bigger distance between two consecutive clusters
D_Max = Max($D_m$) , $0<m<d$
mo = m / $D_{mo}$ = D_Max
$V_{mo}$ = $V_m$ / $D_{mo}$ = D_Max
Border = D_max/2 + $V_{mo}$
//create the node in the space tree with the border value as index and 2 child L_Child and R_Child
CreateNode(Border, R_Child, L_Child)
//Same process with the two child and the resulting regions
CCRight = {$C_i \in$ Sc s.t. $Cc_{imo}$>Border}
CCLeft = {$C_i \in$ Sc s.t. $Cc_{imo}$<Border}
SpaceTree(CCRigth, R_Child)
SpaceTree(CCLeft, L_Child)
// each leaf node points to the B+ tree created by the Pyramid Corresponding to a cluster
CreateLeaf(Pyramid(Sc))
End
```

4. PERFORMANCES

The implementation is achieved in the environment of an application aiming to provide tools to search video sequences in large databases of company films. This environment uses the Java language. The performance evaluation is done on a computer with AMD 3Ghz CPU and 1024MB RAM.
We measure the response time for a Window Query under different workloads: different data set sizes, dimensionality and selectivity. We evaluate performances on synthetic clustered data. In our results, we consider 4 data clusters. We use randomly created queries which follow the data distribution. The response time values are the average of several executions.
Experimental results of our approach show its performance compared to other indexing methods: sequential scan, Pyramid Technique and P+ Tree.
First we vary the data size from 100,000 to 1,000,000 data points. Figure 3 and 4 show the evolution of total response time with respect to data set size in dimension 10 and 24 respectively. It confirms that the Pyramid Technique is not adapted. Our method outperforms sequential scan and P+ tree. Kpyr has a small response time as the processing data set size is smaller to P+Tree's one. This is due to a better clustering method allowing Pyramid Technique application on homogenous data sets.

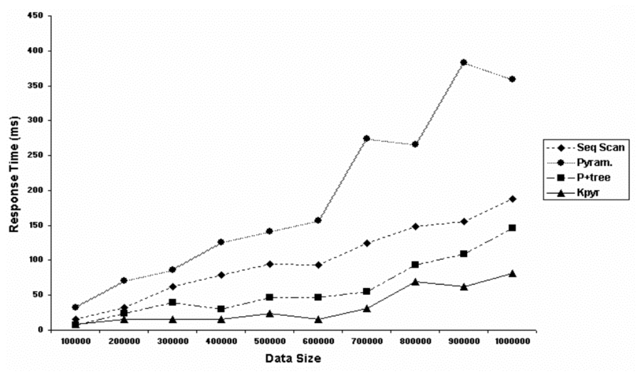

Figure 3. Total Response Time for dimension 10

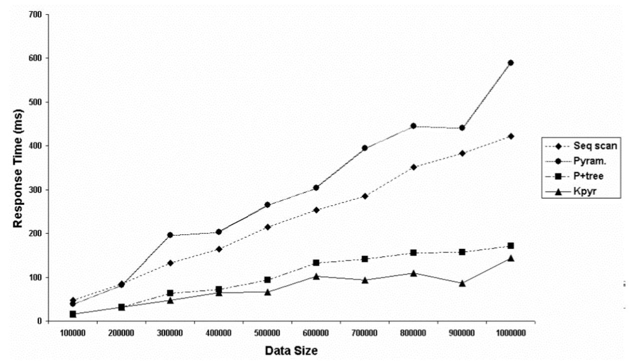

Figure 4. Total Response Time for dimension 24

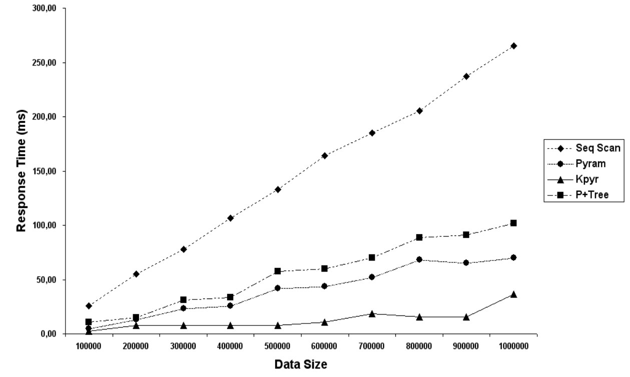

Figure 5. The effect of bad space division in P+tree

In general, the curves of P+Tree are slightly higher than those of Kpyr as shown in figures 3 and 4. However, for some data distributions the performances are worse. Figure 5 shows the curves of results of the example described in section 3.
Some results show that if the WQ size (selectivity) grows, then the performances of P+Tree and Kpyr methods come closer to the sequential scan ones. But, as the queries follow the same distribution as the data, increasing the WQ is meaningless in practice.
The experiments were achieved for WQ. KNN queries are also supported by our approach. We just need to increase gradually the size of the WQ until obtaining the KNN responses. However, it is necessary to store in memory the scanned regions to avoid seeking already found points.

5. CONCLUSION AND FUTURE WORKS

Within the framework of an application aiming to provide intelligent tools of video sequences search in a very large database of company's films, we proposed Kpyr, a new multidimensional indexing technique. Kpyr indexing structure takes into account the fact that the data are classified into topics and that the queries follow these semantics. Our approach adapts the Pyramid Technique for clustered data sets. Experimental results show that our approach outperforms recent existing indexing methods.
The performance evaluation of Kpyr on a larger video database is in progress.

ACKNOWLEDGEMENT

This work is sponsored by RIAM (Innovation Network in Multimedia and Audiovisual) grant of Research French ministry. This work is also the result of strong collaborations of this paper authors and Bruno Bachimont (UTC-Heudiasyc, France), Edouard Gérard, Jean de Bissy (SYLIS FRANCE), Olivier Lombard and Patrick Alleaume (Ouest Audiovisuel, France).

REFERENCES

[1] S. Berchtold, C. Bohm and H.-P. Kriegel, “The Pyramid Technique: Towards Breaking the Curse of Dimensionality”, in Proc. ACM SIGMOD, Seattle, 1998.

[2] B. C. Ooi, K. L. Tan, C. Yu and S. Bressan, “Indexing the Edges - A Simple and Yet Efficient Approach to High-Dimensional Indexing”, in ACM SIGMOD SIGACT SIGART Dallas, Texas, May 2000.

[3] C. Yu, B. C. Ooi, K.-L. Tan and H. V. Jagadish, “Indexing the distance: An efficient method to KNN processing”, in VLDB, pp421-430, Roma, Italy, September 2001.

[4] Rui Zhang, Beng Chin Ooi and Kian-Lee Tan, “Making the Pyramid Technique Robust to Query Types and Workloads”, IEEE Conf. Data Engineering, Boston, 2004.

[5] S. M. Savaresi, D.L. Boley,S. Bittanti and G. Gazzaniga, “Cluster selection in divisive clustering algorithms”, in Proc SIAM Int. Conf. on Data Mining, 2002.

[6] A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching”, Proc. ACM SIGMOD, Boston, 1984.

[7] S. Berchtold, D. A. Keim, and H.-P. Kriegel, “The X-tree: An Index Structure for High-Dimensional Data”, in Proc. VLDB, Bombay, 1996.

[8] D. Comer, “The Ubiquitous B-Tree”, in ACM Computing Surveys 11(4): 121-137, 1979.

[9] J. MacQueen, “Some methods for classification and analysis of multivariate observations”, in Proc. Fifth Berkeley Symp. University of California Press 1, pp. 281-297. 1966.