Topological Transformation Approaches To Database Query Processing

Alok Watve Sakti Pramanik Salman Shahid Chad R. Meiners Alex Liu

Abstract—This paper presents a novel approach that transforms the feature space into a new feature space such that a range query in the original space is mapped into an equivalent box query in the transformed space. Since box queries are axis aligned, there are several implementational advantages that can be exploited to speed up the retrieval of query results using R-Tree [9] like indexing schemes. For two dimensional data the transformation is precise. For larger than two dimensions we propose a space transformation scheme based on disjoint planer rotation and a new type of query, pruning box query, to get the precise results. Experimental results with large synthetic databases and some real databases show the effectiveness of the proposed transformation scheme. These experimental results have been corroborated with suitable mathematical models. In disjoint planer rotation, additional computation time is required to remove the false positives produced due to the bounding box not being precise. A second topological transformation scheme is presented based on optimized bounding box, which reduces the amount of false positives. Amount of this reduction is more with increasing dimensions. Optimized bounding box for higher dimensions is computed based on a novel approach of simultaneous local optimal projections.

Index Terms—Box Query, Range Query, Similarity Search, Query Transformation

1 Introduction

Some of the most common types of queries used in a database system are range queries and box queries. There are numerous application areas for range queries using L_1 space that include Geographical Information Systems, Image Databases and Bioinformatics. In this paper, we focus on the implementation of range queries in L_1 space and use equivalent box queries for optimizing this implementation.

1.1 Background

We begin by formally defining the range query. Let D be the set of d-dimensional records in the database then the range query with radius r at point $p = (p_1, p_2, ...p_d)$, denoted by $r@(p_1, p_2 \dots p_d)$, is defined as,

$$r@(p_1, p_2 \dots p_d) = \{q | q \in D \wedge d(p, q) \leq r\} \tag{1}$$

where, $q = (q_1, q_2, ...q_d)$ is a point in space and $d(p, q)$ is the function giving distance between the point p and the point q. The distant function serves as the measure of dissimilarity between the points. We focus on L_1 distance measure which is formally defined as,

$$d(p, q) = L_1(p, q) = |q_1 - p_1| + \cdots + |q_d - p_d| \tag{2}$$

where, $p_i (1 \leq i \leq d)$ is i^{th} dimension of point p.

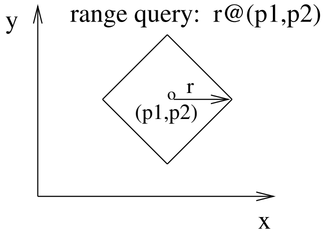


Fig. 1: Range query

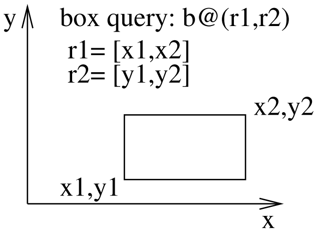


Fig. 2: Box query

For efficient execution of queries in very large databases, usually a multi-dimensional index is created for these records and queries are implemented using this index. Effectiveness of an index for implementing the range query is determined by the number of nodes accessed in the index tree during the execution of the query. We will use R-Tree [9] based indexes (R*-Tree [2] and Packed R-Tree [15]) for efficiently implementing range queries. R-tree type index structures are more suited for implementing box queries than range queries because box queries have similar type of query space (rectangular) as the type of bounded space used for an index node (minimum bounding box) in the R-tree. We, therefore, convert a range query into an equivalent box query first and then implement this box query using R-tree type index structure. A box query with range $r_i = [min_i, max_i]$ in dimension i is defined as follows:

$$b@(r_1, r_2, ..., r_d) = \left\{ q \;\middle|\; \begin{array}{l} q \in D \\ \wedge \\ min_i \leq q_i \leq max_i \text{ for } 1 \leq i \leq d \end{array} \right\} \tag{3}$$

Figure 1 gives an example of a range query and Figure 2 gives an example of a box query.

- *Research was partially supported by NSF grant no. 1319909*
- *A. Watve, S. Pramanik, Salman Shahid and A. Liu are with the Department of Computer Science and Engineering, 3115 Engineering Building, Michigan State University, MI 48824-1226, USA*
  Email : {watvealo, pramanik, shahids1, alexliu}@cse.msu.edu
- *Chad R. Meiners is currently with MIT Lincoln Lab. The proposed work was done when he was a PostDoc at the Department of Computer Science and Engineering, 3115 Engineering Building, Michigan State University, MI 48824-1226, USA*
  Email : meinersc@cse.msu.edu

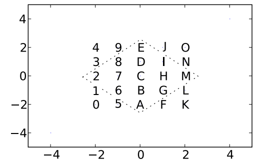


Fig. 3: Range query

⇔

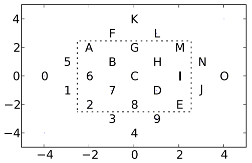


Fig. 4: Transformed query

1.2 Converting range queries into box queries

Implementing range queries using indexing is a well studied problem. However, to the best of our knowledge, transforming a range query into an equivalent box query and then implementing the box query on R-tree type indexes has not been studied before. Our motivation for this transformation is that in L_1 space, range query is a $d-$dimensional hyper-diamond which can be closely approximated by a box query. In fact, in 2-dimension this conversion is exact. Thus, we propose transforming data space so that a range query becomes a box query. Once we find such a transformation, we can implement box query by applying any existing indexing scheme that uses bounding boxes for data space partitioning.

We illustrate the transformation in 2-dimension by using an example range query $2@(0,0)$, shown by the dotted diamond in Figure 3. Each point in the figure is represented by a single character so that we can identify the same point in the transformed space shown in figure Figure 4. The range query $2@(0,0)$ enclosing the set of points $\{2,6,7,8,A,B,C,D,E,G,H,I,M\}$ is transformed into a minimal bounding box query $b@([-2,2],[-2,2])$ (dotted rectangle) of Figure 4 enclosing the same set of points. Note that points that are not inside the diamond are also not inside the rectangle. Actual transformation function to achieve this homology in the transformed space is linear and is given in subsection 3.1.

The transformation is more challenging in the case where the number of dimensions is greater than two. We tackle this problem by transforming two dimensional projections of the space. As an artifact of this, the bounding box no longer models the range query exactly, creating some false positives. We propose a novel pruning method to alleviate this problem. However, with increasing dimensions the number of false positives increases. Removing these false positives incur additional main memory computation. We give another scheme requiring fewer number of false positives based on optimized bounding box derived from using the concept of Simultaneous Local Optimal (SLO) projections.

1.3 Key contributions of the paper

- We propose a novel multi-dimensional space transformation scheme to convert range queries using L_1 distance into equivalent box queries. We then show that implementing box queries in the transformed space provide better performance than existing methods for implementing range queries that use R-tree type index structures.
- We provide detailed theoretical analysis to estimate various performance trade-offs resulting from the proposed transformation. Experimental results corroborate our theoretical analysis.
- We demonstrate that the proposed transformation scheme is effective even for the k-nearest neighbor queries. We show that the performance of the proposed scheme for KNN search is better than those commonly used for KNN search using R-tree type index structure.
- We present a novel algorithm, based on Simultaneous Local Optimal projections, to compute optimized bounding box for a given set of points in n-dimension, on a convex hull. Using this SLO approach, we perform space transformation to convert range queries into box queries. We show that the proposed algorithm provides better performance than those commonly used for range queries using R-tree type index structures. Amount of this performance gain gets significantly better with increasing dimensions.

Some of the initial work presented here is published in [25].

1.4 Outline of the paper

The rest of the paper is organized as follows: In section 2, we present the prior work related to this paper. Section 3 proposes a transformation method for 2-D data. These concepts are extended for higher dimensional data in section 4. A theoretical model for the proposed transformation is presented in section 5. We discuss the effect of index stability in section 6 followed by experimental evaluation in section 7. Performance of KNN queries in the transformed space is given in section 8. Section 9 describes the algorithm to minimize false positives based on SLO. Concluding remarks follow in the last section.

2 Related Work

There is a significant amount of existing work on implementing range queries and nearest neighbor queries using database indexes. A detailed discussion on these topics can be found in [13], [14]. Existing multidimensional index structures such as the R-Tree [9], X-Tree [4], G-Tree [18] and VA-File [32] have been used for optimizing range searches but these range searches are not specifically optimized for L_1 space. R-tree paper, for example, used search rectangles to evaluate query performance in L_2-space and these queries are not range queries but similar to box queries as defined in subsection 1.1 of this paper. There are indexing schemes that use bounding spheres [6], [33] or sphere/rectangle [16] for their indexing schemes. These indexing schemes are effective for implementing range queries in generic metric spaces and not specifically optimized for L_1-space.

The idea of transforming one data-space into some other data-space for efficient query processing has been around for some time. A detailed discussion about transformation based data access methods can be found in [8]. Some of these methods [7], [11], [17] transform objects (or polygons) into higher dimensional points and then use one of the established point access methods such as the grid file [22]. The others transform the multi-dimensional data to 1-dimensional data using space filling curves (such as z-curve [21] or Hilbert curve

[10]). Such methods employing z-curve for executing box queries are proposed in [23], [29]. Unlike these methods which map a high dimensional point to a one dimensional point, some transformations reduce the dimensionality of the data using methods such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) [1], [5], [26], [31]. The primary motivations for reducing the number of dimensions are to avoid dimensionality curse [3] and to minimize redundancy. Focus of this work is not on reducing dimension of the data set but to achieve range query optimization for for any given dimension.

Linear transformation is a well known technique in linear algebra [19]. However, its applications to database queries have not been explored much. To the best of our knowledge, other than our work in [25] , there is no other work on mapping of range queries to box queries using data transformation. The proposed work focuses on using linear transformation within the framework of existing dynamic database indexes for improving page accesses for range queries. It also guarantees 100% recall and scalability with increasing database size thus overcoming some of the disadvantages of prior work.

3 2-D TRANSFORMATIONS

Mapping range queries into box queries requires transformations of both data space and user queries. The space transformation is performed offline on the data, before building the index. For simplicity, we call a database built with a transformed data a "transformed database". The query transformation is performed online on each query. The transformation needs to satisfy the property that the result of the transformed query over the transformed database is equal to, or at least a superset of, the result of the original query over the original database. When the two query results are equal, we call such transformations *precise transformations*; otherwise, we call them *approximate transformations*. Approximate transformation may introduce false positives (*i.e.*, the points that do not satisfy the original query but do satisfy the transformed query) or false negatives (*i.e.*, the points that are in the original query but are not in the transformed query). Precise transformations have neither false positives nor false negatives. We now present the proposed transformations. It will be shown later that our transformations are precise.

3.1 Transformation Function

In this section, we present the transformations from range queries to box queries for 2 dimensional (2-D) databases.

3.1.1 Space Transformation:

For 2 dimensional (2-D) databases, our transformations from range queries to box queries are accomplished by mapping each point (x, y) in the original 2-D space to the point $(x + y, x - y)$ in the transformed space, which is essentially a change of axis as shown in Figure 3.

Formally, our transformation function $T\colon \mathbb{R}^2 \to \mathbb{R}^2$ is defined as,

$$T(x, y) = (x + y, x - y) \tag{4}$$

And the inverse transformation function $T^{-1}\colon \mathbb{R}^2 \to \mathbb{R}^2$ is defined as,

$$T^{-1}(x, y) = (\frac{x + y}{2}, \frac{x - y}{2}) \tag{5}$$

This function essentially changes the two axes so that they align perfectly with the geometric faces of range queries in the original space. By such transformations, a range query in the original space based on L_1 distance becomes a box query in the transformed space. For any 2-D database D, which is a set of 2-D points in the original space, the transformed database $T(D)$ is defined by $T(D) = \{T(x, y)|(x, y) \in D\}$.

Note that we do not need to store $T(D)$ as a separate database; rather, we build an index for $T(D)$, which points back to the original points in D.

3.1.2 Query Transformation:

Mathematically, $r@(a, b)$ denotes the set of all the points that are within range r based on L_1 distance from point (a, b). Geometrically, all the points in $r@(a, b)$ form a diamond with four end points: $(a + r, b)$, $(a, b + r)$, $(a - r, b)$, $(a, b - r)$. All points in $b@([a_1, b_1], [a_2, b_2])$ form a rectangle with four end points: (a_1, b_1), (a_2, b_1), (a_2, b_2), (a_1, b_2).

The space transformation transforms the four corners of the range query $r@(a, b)$ to a square with four end points: $(a+r+b, a+r-b)$, $(a-r+b, a-r-b)$, $(a+r+b, a-r-b)$, $(a - r + b, a + r - b)$. Thus, these transformed points are precisely the representation of a box query $b@([a + b - r, a + b + r], [a - b - r, a - b + r])$ in the transformed space. Geometrically, our 2-D transformation from range queries to box queries converts a diamond in the original space to a square in the transformed space. For example, in Figure 3, range query $2@(0, 0)$ in the original space is equivalent to the box query $b@([-2, 2], [-2, 2])$ in the transformed space. Note that, in the discussion so far, we implicitly assume that the database objects are points in 2-D space. However, it can be seen that the proposed transformation can be applied to indexing other object structures such as lines/planes or boxes. Real world applications (such as map databases) may use bounding rectangles to approximate lines and other objects, which are then indexed using R-Tree like index structures. Extension of this technique for these objects is straightforward. A rectangle in the original space is transformed by simply applying the transformation function to each of its vertices. This transformation is similar to query transformation and it can be easily seen that the object formed by these transformed point is a rectangle in transformed space.

3.2 Transformation Properties

We present several important properties of our transformation function T defined in formula 4.

3.2.1 Precision Property

Theorem 3.1 shows that both our range to box query transformation and box to range query transformation are precise.

Theorem 3.1: For any point $(x, y) \in D$, (x, y) satisfies the range query $r@(a, b)$ if and only if (iff) $T(x, y)$ satisfies the box query $b@([a + b - r, a + b + r], [a - b - r, a - b + r])$.

Proof: For any two numbers u and v, $|u| + |v| \leq r$ iff $|u+v| \leq r$ and $|u-v| \leq r$. Based on this, $|x-a|+|y-b| \leq r$ holds iff both $|(x+y)-(a+b)| = |(x-a)+(y-b)| \leq r$ and $|(x-y)-(a-b)| = |(x-a)-(y-b)| \leq r$ hold. Note that (x, y) satisfies the range query $r@(a, b)$ iff $|x-a|+|y-b| \leq r$ holds, and $T(x, y)$ satisfies the box query $b@([a+b-r, a+b+r], [a-b-r, a-b+r])$ iff $|(x+y)-(a+b)| \leq r$ and $|(x-y)-(a-b)| \leq r$ holds. □

3.2.2 Distance Property

The proposed transformation function T does not preserve L_1 distance, even though it is precise.

Consider two points $p = (1, 1)$ and $q = (1.5, 0)$. Distance of these points from origin is 2 units and 1.5 units respectively. Using the transformation, $T(p) = (2, 0)$ and $T(q) = (1.5, 1.5)$. As origin is unaffected by the transformation, the distances of transformed points from the (transformed) origin are 2 units and 3 units respectively. It can be seen that neither the distance nor the relative ordering of points is preserved.

Theorem 3.2: For any two points (x_1, y_1) and (x_2, y_2), $L_1(T(x_1, y_1), T(x_2, y_2)) =$
$L_1((x_1, y_1), (x_2, y_2)) + ||x_1 - x_2| - |y_1 - y_2||$.

Proof: For any two numbers u and v, $|u + v| + |u - v| = |u| + |v| + ||u| - |v||$. Based on this fact, we have $L_1(T(x_1, y_1), T(x_2, y_2)) = L_1((x_1 + y_1, x_1 - y_1), (x_2 + y_2, x_2 - y_2)) = |(x_1 + y_1) - (x_2 + y_2)| + |(x_1 - y_1) - (x_2 - y_2)| = |(x_1 - x_2) + (y_1 - y_2)| + |(x_1 - x_2) - (y_1 - y_2)| = |x_1 - x_2| + |y_1 - y_2| + ||x_1 - x_2| - |y_1 - y_2|| = L_1((x_1, y_1), (x_2, y_2)) + ||x_1 - x_2| - |y_1 - y_2||$. □

We can prove a similar property for T^{-1}.

Corollary 3.1: For any two points (x_1, y_1) and (x_2, y_2), $L_1(T^{-1}((x_1, y_1)), T^{-1}((x_2, y_2)))$
$= (L_1((x_1, y_1), (x_2, y_2)) + ||x_1 - x_2| - |y_1 - y_2||)/2$.

3.2.3 Inequality Property:

Although transformation function T does not preserve L_1 distance, Theorem 3.3 shows an important special case where the transformation function T preserves distance inequality.

Theorem 3.3: Given a point (x_1, y_1), an MBR represented as a rectangle B, and a point (x_2, y_2) on the edge of the rectangle, if among all the points in B, (x_2, y_2) is the point that is closest to (x_1, y_1), then $T((x_2, y_2))$ is the closest point in $T(B)$ to $T((x_1, y_1))$ and $T^{-1}((x_2, y_2))$ is the closest point in $T^{-1}(B)$ to $T^{-1}((x_1, y_1))$

Proof: Our proof is based on the fact that for any four non-negative numbers u, v, w, and z, if $u+v \leq w+z$, $u \leq w$, and $v < z$, then $u + v + |u - v| \leq w + z + |w - z|$. We omit the proof of this fact.

For any two points (x_1, y_1) and (x_2, y_2), we use $L_1((x_1, y_1), (x_2, y_2))$ to denote their L_1 distance. Considering any point (x_3, y_3) in the rectangle, because (x_2, y_2) is closer to (x_1, y_1) than (x_3, y_3), we have $L_1((x_2, y_2), (x_1, y_1)) \leq L_1((x_3, y_3), (x_1, y_1))$, $|x_2 - x_1| \leq |x_3 - x_1|$ and $|y_2 - y_1| \leq |y_3 - y_1|$. Now we need to prove $L_1(T((x_2, y_2)), T((x_1, y_1))) \leq L_1(T((x_3, y_3)), T((x_1, y_1)))$. By Theorem 3.2, we have $L_1(T((x_2, y_2)), T((x_1, y_1))) = L_1((x_1, y_1), (x_2, y_2)) + ||x_1 - x_2| - |y_1 - y_2|| = |x_1 - x_2| + |y_1 - y_2| + ||x_1 - x_2| - |y_1 - y_2||$ and $L_1(T((x_3, y_3)), T((x_1, y_1))) = L_1((x_1, y_1), (x_3, y_3)) + ||x_1 - x_3| - |y_1 - y_3|| = |x_1 - x_3| + |y_1 - y_3| + ||x_1 - x_3| - |y_1 - y_3||$. By the above fact, we have $|x_1 - x_2| + |y_1 - y_2| + ||x_1 - x_2| - |y_1 - y_2|| \leq |x_1 - x_3| + |y_1 - y_3| + ||x_1 - x_3| - |y_1 - y_3||$. □

4 Multi-dimensional Transformations

For $d = 2$, range queries can be precisely transformed into box queries. However, to the best of our knowledge, for $d > 2$ we have not found any work in the literature on precise transformations. We conjecture that such a precise transformation may not exist. Since we cannot define such a precise transformation, we develop a new type of box query that uses the range value from the original range query to prune the false positives in the transformed box query while preventing the occurrence of false negatives.

In this section, we first use the 2-D transformation of Section 3 to define an approximate transformation called disjoint planar rotations (DPR). Second, we describe how to use the disjoint planar rotations to map a range query into a precise box query.

4.1 Disjoint Planar Rotations

DPR is a transformation that is derived from our two dimensional transformation function. We transform a d dimensional space via this technique by transforming disjoint planes in the database. For example, a four dimensional point (x, y, z, w) can be transformed into $(T(x, y), T(z, w))$. That is, this transformation can be visualized as a rotation of each disjoint plane in the database's space.

More formally, we define a d dimensional transformation $T^d(p)$ as follows:

$$T^d(p) = \begin{cases} \begin{pmatrix} T(p_1, p_2), T(p_3, p_4), \\ \cdots, \\ T(p_{d-1}, p_d) \end{pmatrix} & \text{when } d \text{ is even} \\ \begin{pmatrix} T(p_1, p_2), T(p_3, p_4), \\ \cdots, \\ T(p_{d-2}, p_{d-1}), p_d \end{pmatrix} & \text{when } d \text{ is odd} \end{cases} \tag{6}$$

Note that in the odd case we choose to preserve the last dimension because in 1-dimensional space, range query and box query are one and the same and they differ only in the representation (e.g. $r@(x) \Leftrightarrow b@([x - r, x + r])$), which obviates the need for an explicit transformation.

4.2 Pruning Box Query

Our modification to box query (which we call Pruning Box Query) is based on the observation that if we can estimate distance between the query center and an MBR of the index tree, we can prune the branches of the tree that do not contain any true positives. We first prove the result proposed in theorem 3.3 for $d-$dimensional data.

Theorem 4.1: Given a point c and a set of points B representing the points from an MBR. If p is the closest point in B to c, then $T^d(p)$ is the closest point in $T^d(B)$ to $T^d(c)$ and $(T^{-1})^d(p)$ is the closest point in $(T^{-1})^d(B)$ to $(T^{-1})^d(c)$.

Proof: The proof is similar to that of theorem 3.3 and is omitted. □

Based on the theorem we propose following heuristic to eliminate all the false positives:

Heuristic: If an MBR M overlaps with the query box, we find the closest point $T^d(p)$ in M to query center $T^d(c)$. Using the inverse transformation we then calculate distance between p and c; if it is greater than the query range then the MBR is pruned.

Let u be the point in MBR M that is nearest to center of the box query b. Using the above heuristic, we now formally define pruning box query (PBQ) as,

$$pbq@(r_1, r_2, ..., r_d) = \left\{ q \;\middle|\; \begin{array}{l} q \in D \\ \wedge \;\; (min_i \leq q_i \leq max_i \\ \;\text{for } 1 \leq i \leq d) \;\; \wedge \\ \text{distance between } (T^{-1})^d(q) \\ \;\text{and } (T^{-1})^d(u) \\ \text{is less than the query radius} \end{array} \right\} \tag{7}$$

In equation 7 above, $r_i = [min_i, max_i]$ is calculated using the method for computing the query bounds in section 3.1.2. If the number of dimensions is odd *i.e.* when the last dimension c_d of point c is not transformed, r_d is calculated as, $r_d = [c_d - r, c_d + r]$, r being the query radius. This approach not only eliminates all the false positives but it also provides more efficient query execution. It is important to note that we do not miss any data record with this pruning strategy. Theorem 4.2 states this.

Theorem 4.2: For every point p that satisfies the range query $r@(c_1, c_2 \ldots c_d)$, p is also contained in the result of the PBQ.

Proof: Let $c = (c_1, c_2 \ldots c_d)$. Further, Let, if possible, there be a point p such that, $p \in r@(c_1, c_2 \ldots c_d)$ but p' is not contained in the box query, where $p' = T^d(p)$. This is possible, only if the MBR M containing p' was pruned by the box query at some point, i.e. the estimated distance between M and c' ($c' = T^d(c)$) was greater than r. Let $u' = T^d(u)$ be the closest point in M to c'. This implies that while u' was the closest point to c' in the transformed domain, u was not the closest point to c in the original data domain. This contradicts theorem 4.1. Hence, such a point p does not exist. In other words, resultset returned by the PBQ contains all the points from the one returned by the original range query. □

4.3 Search algorithm

Based on the concepts developed so far, we summarize our search algorithm using PBQ in algorithm 1.

5 Theoretical Analysis

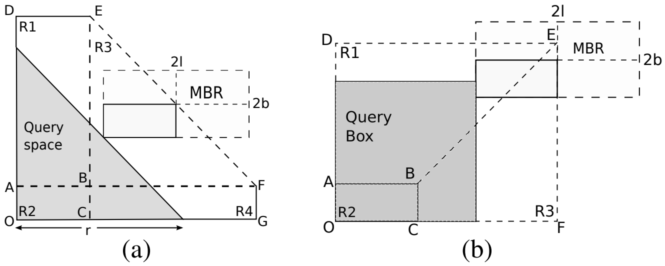
Using DPR and pruning box queries improves the performance of indexed queries because the transformation aligns the index's minimum bounding boxes' faces with faces of the query. In this section we provide a detailed analysis of the range query and the PBQ performances. We first present it for 2-dimensional queries and then generalize it for higher dimensions.

Algorithm 1 PBQ Search

```
Input: Query point $q$, query radius $r$ and root node $N_r$ of the
    search tree (built in transformed space).
  Transform $q$ to $T^d(q)$ using DPR in equation 6
  $B_q \leftarrow$ Query box in the transformed space.
  Result set $R \leftarrow \{\}$
  Add $N_r$ in to the queue $NQ$ of nodes to search.
  while $NQ$ is not empty do
      Get the next node $N$ from $NQ$.
      if $N$ is leaf node then
          Add all the points in $N$ satisfying the query to $R$.
      else
          for Each child node $N_c$ in $N$ do
              if $B_q$ intersects with MBR of $N_c$ then
                  Find the point $T^d(p)$ in MBR of $N_c$ closest
                  to $T^d(q)$
                  if $L_1(p, q) < r$ then
                      Add $N_c$ in $NQ$.
                  end if
              end if
          end for
      end if
  end while
  return $R$
```

5.1 Model Basics

Without loss of generality, we fix the size of all MBRs so that we can calculate the area of MBR centroids whose corresponding MBRs intersect with a query. From this area, we can calculate the probability that an MBR of a certain size will intersect with a query. For example, for 2-D query spaces, we fix an MBRs length to be $2l$ and breadth to be $2b$. We must calculate probability that a random MBR of certain size intersects the query space (a diamond in case of the range query and a square for the PBQ). We analyze only one quadrant of the 2-D plane. Analysis for other quadrants is similar and is omitted.



(a) (b)

Fig. 5: The MBR intersection areas in a quadrant for range and transformed range queries

Figure 5 shows the space in which an MBR of size $2l \times 2b$ must lie in order to intersect with the query space. We can see from this visualization that the query faces align with the MBR faces after transformation, and we conjecture that this alignment improves query performance for two reasons: MBRs are less likely to intersect with the PBQ than the range query, and MBRs that do intersect with PBQ have a higher likelihood of containing a point within the query than MBRs that intersect the range query. We would like to clarify here that, only the data space is transformed, not the MBRs. So MBR in figure 5 (b) does not correspond to MBR in (a). In this analysis, we simply compute and compare the probability that an MBR of certain size intersects with the query in the original and transformed space.

5.2 Analysis of Range Query

To calculate the probability of intersection for a range query $P_{R(r,l,b)}$ with an MBR, we determine the area in which the centroid of an MBR with length $2l$ and breadth $2b$ must lie for it to intersect with the query. As shown in Figure 5(a), the intersection space can be divided into four regions $R_1(\square ADEB), R_2(\square OABC), R_3(\triangle BEF)$ and $R_4(\square CBFG)$. The area of the intersection space can then be calculated as,

$$\begin{aligned} A_{R(r,l,b)} =& \text{Area of R1, R2, R3, and R4} \\ =& \int_0^r b\ dx + lb + \int_0^{\frac{r}{\sqrt{2}}} 2x\ dx + \int_0^r l\ dx \\ =& \frac{r^2}{2} + lb + r(l+b) \end{aligned} \tag{8}$$

Hence, given the area of the data space, A, the probability that an MBR will overlap with a range query is,

$$P_{R(r,l,b)} = \frac{A_{R(r,l,b)}}{A} \tag{9}$$

5.3 Analysis of Box Query

To calculate the probability $P_{B(r,l,b)}$ of intersection of a PBQ with an MBR, we determine the area in which the centroid of an MBR with length $2l$ and breadth $2b$ must lie for it to intersect with the query. As shown in Figure 5(b), the intersection space is a rectangle with length of $2(l+r)$ and breadth of $2(b+r)$. In other words, any MBR whose center lies in the $\square ODEF$ will intersect with the box query. Note that the space has dilated by a factor of 2 due to the transformation. Hence, the total area of the space in the transformed domain is $2A$. We divide the space into three regions $R_1(\square ADEB)$, $R_2(\square OABC)$ and $R_3(\square CBEF)$. The area of the intersection space can then be calculated as,

$$\begin{aligned} A_{B(r,l,b)} =& \text{Area of R1, R2, and R3} \\ =& \int_0^r (l+x)\ dx + lb + \int_0^r (b+x)\ dx \\ =& r^2 + lb + r(l+b) \end{aligned} \tag{10}$$

Hence, given the area of the data space, $2A$, the probability that a random MBR intersects the transformed range query is,

$$P_{B(r,l,b)} = \frac{A_{B(r,l,b)}}{2A} \tag{11}$$

5.4 Hyper-dimensional Queries

Figure 6 represents the four dimensional space as a two dimensional grid of two dimensional slices through the four dimensional space. In this case, the grid is a coarse grain visualization of the effect of the wz plane on the xy planes so each panel in the wz represents the xy plane that is fixed at the wz panel's coordinate. The visualization is easier to understand if we note that in L_1 space, query radius in 4-D space is sum of query radii in xy plane and that in wz plane. For a point at a distance r from the query, as its distance in xy projection increases, its distance in wz plane must decrease. Hence, in Figure 6, a large projection in xy plane corresponds to small projection wz plane and vice versa. While this visualization is coarse grained in that it does not show every point in the range query, it illustrates how DPR transforms the range query into a PBQ as shown in Figure 6(b), which shows Figure 6(a)'s range query as a PBQ in a DPR transformed space. Note that this visualization can be generalized to visualize any hyperspace as a nested series of two dimensional grids. with more than two dimensions. The above equations can be generalized to any even number of dimensions via a density function. We use this nesting concept to find the area of centroids for any even dimensional query. As we move away from the center of the query in xy plane, density of points (or query space) in wz plane decreases. We define density function as the area in which center of an MBR must lie in order to intersect with query space. The d-dimensional density function for range query is denoted as $A_{R(r,W,d)}$ and that for the box query is denoted as $A_{B(r,W,d)}$.

Consider the MBR in Figure 6(a); we first examine the intersections of the wz projections of the MBR and query, which is shown by the dotted lines. Note that if these projections did not intersect there would be no intersection of query and MBR; however, since there is an intersection we can determine if query and MBR do intersect by looking for an intersection in the xy projection that is closest to the origin of the range query.

Given a hyper-rectangle with widths $W = \langle w_1, \cdots, w_d \rangle$, the density function for the range query is recursively defined as,

$$\begin{aligned} A_{R(r,W,0)} =& 1 \\ A_{R(r,W,d)} =& \int_0^r A_{R(r-x,W,d-2)} w_d\ dx \\ &+ A_{R(r,W,d-2)} w_{d-1} w_d \\ &+ \int_0^{\frac{r}{\sqrt{2}}} A_{R(r-x\sqrt{2},W,d-2)} 2x\ dx \\ &+ \int_0^r A_{R(r-x,W,d-2)} w_{d-1}\ dx \end{aligned} \tag{12}$$

$$P_{R(r,W,d)} = \frac{A_{R(r,W,d)}}{A} \tag{13}$$

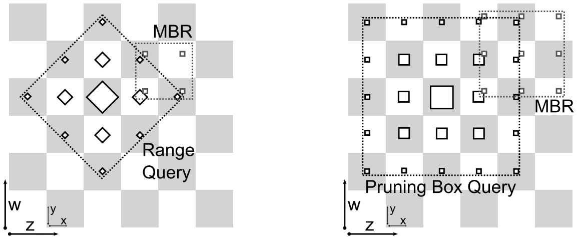
in the original space and

$$
\begin{aligned}
A_{B(r,W,d)} =& 1 \\
A_{B(r,W,d)} =& \int_0^r A_{B(r-x,W,d-2)}(w_{d-1}+x)\ dx \\
&+ A_{B(r,W,d-2)} w_{d-1} w_d \\
&+ \int_0^r A_{B(r-x,W,d-2)}(w_d+x)\ dx \qquad (14) \\
P_{B(r,W,d)} =& \frac{A_{B(r,W,d)}}{2^{d/2}A} \qquad (15)
\end{aligned}
$$

in the transformed space. It can be seen that area analyses for two dimensional cases are in fact special cases of equations 12 through 14. Space dilation mentioned in the previous section, is responsible for the factor $2^{d/2}$ in the denominator. Space dilation also causes the bounding boxes in transformed space to be bigger. Thus, the probability of intersection does not decrease exponentially with dimensions. However, as shown in the results section, the performance improvement increases significantly with dimensions.

It can be shown that the density function for the PBQ is less than the density function for the range query, when we increase the number of dimensions. Hence, the range query has a larger number of valid centroids that intersect it than the pruning box query.

5.5 Performance improvement



(a) (b)

Fig. 6: Visualizations of range and pruning box queries relationship with MBRs in hyperspace

We define relative improvement in performance due to the proposed transformation as the ratio of query IO of a transformed box query to that of original range query. For uniformly distributed data and uniformly distributed query points probability of intersection of MBR with query point derived above can be used as an estimate of expected IO. We would like to highlight that our analysis applies for a given level in the tree. Leaf level heavily dominates IO required for any query, hence, we can use IO estimated at the leaf level as a good approximation of query IO. Hence we can use equations 13 and 15 to estimate expected relative improvement due to the transformation as,

$$I(r,W,d) = P_{B(r,W,d)}/P_{R(r,W,d)} \qquad (16)$$

When range r is considerably large compared to size of the MBRs, the first term in equations 8 and 10 will dominate and we will not have significant improvement. However, in most of the real world systems, query range is usually small so that it retrieves few records. Hence, based on equations 9 and 11, we can expect a considerable performance gain. Performance gain based on the actual values of b and l in the original as well as the transformed space is given in section 7.

6 Effect of Index Structure on Performance Improvement

Splitting decisions in R*-Tree are made based on the points seen so far, hence, its node splits are heavily dependent on the data. For two databases drawn from the same data distribution, R*-Tree can be considerably different. In fact even for the same database, performance of R*-Tree can be very different for different insertion order. This fact is particularly evident in higher dimensional spaces. We have observed that the performance improvement due to transformation is sensitive to the structure of the R*-tree being built. We use average leaf diameter to assess goodness of an index structure. Leaf diameter is defined as the maximum distance between a pair of points in a leaf node. Leaf diameter measures compactness of a leaf node. Smaller leaf diameter implies that the points within the leaf are close to each other (hence are more similar to each other). An index with smaller leaf diameter does a better job partitioning the data space into index pages and is expected to have better performance. Our experiments show a very high correlation between leaf diameter and query IO. Further, we observed that performance improvement due to the transformation is better when the average leaf diameter is smaller. Average leaf diameter varies with the order in which data set is being inserted into the tree. As a result, we see a wide variation in performance improvement with different insertion orders.

Tables 1 and 2 summarize our findings. For all the experiments, database size was fixed at 10 million records. We used 10 different insertion orders within the same database for the first experiment and 10 different databases, having the same distribution but generated using different seeds, for the second. Average number of page accesses and standard deviation of the number of page accesses was calculated. It can be seen from these tables that there exists a large variance in query performance of R*-Tree. There are cases where performance improvement due to the transformation is observed to be in excess of 40%.

Query type	# Dimensions	Mean IO	Std. Deviation
RQ	2	27.67	0.21
	10	7678.05	1626.54
PBQ	2	25. 09	0.46
	10	7609.86	1465.36

TABLE 1: Mean and Std. deviation of # page accesses in an R*-Tree averaging over various insertion orders

The inherent dependence of R*-Tree on insertion order and statistical properties of the database makes it a poor test-bed for effectively measuring the performance improvement due to transformation. We apply the proposed transformation on a static index called packed R-Tree [15]. Packed R-Tree (and any static index) assumes that all the data is available at the time of indexing. This allows for better tuning of the index. Packed

Query type	# Dimensions	Mean IO	Std. Deviation
RQ	2	27.77	0.35
	10	7220.27	1702.33
PBQ	2	24.88	0.35
	10	6509.47	1375.35

TABLE 2: Mean and Std. deviation of # page accesses in an R*-Tree averaging over various databases

R-Tree sorts the data in Hilbert order before inserting. This creates more compact R*-tree with reduced leaf diameters. Thus, it does not matter which order the data comes in, it will always be indexed in the same order. Further, Packed R-Tree also guarantees (almost) 100% space utilization which considerably reduces the total number disk pages required for the index.

Similar to the R*-Tree we measured mean and variance of number of page accesses for 10 different databases. Table 3 shows results of these experiments. It can be seen that standard deviation in the number of page accesses for a packed R-Tree is very small highlighting its stability. At the same time performance gain in terms of page accesses is significant even for higher dimensions.

Query type	# Dimensions	Mean IO	Std. Deviation
RQ	2	29.21	0.14
	10	3212.96	3.28
PBQ	2	26.41	0.1
	10	2500.88	2.21

TABLE 3: Mean and Std. deviation of # page accesses in a packed R-Tree averaging over various databases

Based on these results, we decide to use Packed R-Tree for all our experiments in order to nullify any positive or negative effect of the index properties and to accurately measure performance improvement due to transformation alone.

7 EXPERIMENTAL RESULTS

In this section we present the results of applying the proposed transformation on various databases. Effectiveness of the proposed transformation is measured by comparing the IO cost (i.e. number of index page accesses) for the proposed pruning box queries with that of range queries. For performance comparison purposes, we create packed R-Tree indexes for the range query in the original space and the pruning box query in the transformed space. It should be noted that the proposed transformation can be used on any R-Tree based index.

Uniformly distributed synthetic datasets as well as some real datasets were used for the experiments. Data records were normalized to unit (hyper)cube. Page size of 4K bytes was used for the index nodes. All results presented here are based on averaging the IO of one hundred random queries. All the experiments were run on AMD Opteron 2.2 GHz systems running GNU/Linux. The labels used for various methods in the figures and tables are as follows: RQ - traditional range query on packed R-Tree, PBQ - Pruning Box Query on packed R-Tree. We include results for linear scan (labeled 10% Linear) wherever they are comparable. However, wherever linear scan is much worse than indexing, it is dropped from the graph for better presentation.

A detailed analysis of effect of database sizes, number of dimensions and query radius is presented in subsequent sections. We first prove the correctness of our theoretical analysis by comparing improvement predicted by our model with the actual improvement observed experimentally.

7.1 Improvement estimation

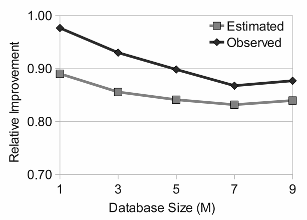


Fig. 7: 2-dimensional data

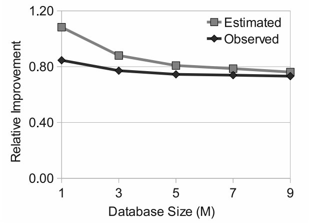


Fig. 8: 10-dimensional data

Comparison of estimated and observed improvement

Figures 7 and 8 compare the estimated values of relative improvements (equation 16) with the observed values for a fixed query for 2 and 10 dimensional databases respectively. For ease of evaluating integrals, we assumed that all MBRs are uniform (*i.e.* $l = b$). It can be seen from the graphs that our analysis is fairly accurate especially for larger databases. When database size is small, some of our assumptions such as MBRs are uniform and they all have the same size, may not be valid. These results are in slight disagreement between observed performance improvement and predicted one. However, as database size increases, the assumptions hold and we see much better agreement in the two values.

7.2 Avoiding Empty Pages

Experimentally, we can see that the reduction in area that our model predicts for pruning box queries translates to fewer page accesses in the index tree. Table 4 shows the performance break down at each level of the index tree for both range and pruning box queries for two and ten dimensions and database of 10 million. The break down shows the average number of page accesses, the average number of empty page accesses, and the average number of non-empty page accesses. An *empty* page has no children that satisfy the query, while a *non-empty* page has at least one child that satisfies the query. For 10-dimensional data, the tree has 4 levels, so in the table, we denote middle two levels as Middle-h (higher level closer to the root) and Middle-l (lower level closer to the leaf).

We observe that both range and pruning box queries have a similar number of non-empty page accesses, which corresponds in our model to the shared centroid area with both queries. The number of non-empty pages access should be similar between both query types because our transformation does not change the relative distribution of the records in the space.

We observe that the performance improvements are best gained by reducing the number of empty page accesses. For

$D = 2$ $R = 0.005$

	PBQ			RQ		
Level	Total	Empty	Non-empty	Total	Empty	Non-empty
Top	1.47	0.44	1.03	1.34	0.31	1.03
Middle	3.97	2.34	1.63	3.46	1.67	1.79
Bottom	20.13	3.35	16.78	23.34	5.44	17.9

$D = 10$ $R = 0.07$

	PBQ			RQ		
Level	Total	Empty	Non-empty	Total	Empty	Non-empty
Top	1.91	0.53	1.38	1.95	0.56	1.39
Middle-h	27.76	22.31	5.45	34.17	29.81	4.36
Middle-l	269.56	255.75	13.81	346.53	335.74	10.79
Bottom	1948.66	1922.56	26.1	2690.01	2665.98	24.03

TABLE 4: Break down of page accesses

example, when $D = 2$, there are very few empty pages, and we see a small difference in performance between queries. However, when $D = 10$, empty pages make up the majority of page accesses, and we see that the pruning box query loads approximately two third the number of empty pages than the range query, which results in a much larger performance improvement.

7.3 2D Transformations

As explained earlier, transformation in 2-dimensional databases is perfect, i.e., the transformed query does not lose any useful results nor does it gather any unwanted results.

7.3.1 Effect of database size

Figure 9 shows the effect of database size on the number of page accesses. As seen from the figure, as the database size increases, number of page accesses for both range and pruning box queries increases, as expected. However, rate of increase for range query is higher than that of the pruning box query. The performance improvement increases with increasing database size. The relatively low improvement is consistent with our analysis in section 7.2.

7.3.2 Effect of query ranges

We experimented with various query ranges keeping database size constant (10 million records). The performance comparison of pruning box query with range query is shown in figure 10. Ranges in the figure are a normalized distances. It can be seen from the figure that pruning-box queries perform consistently better than range queries.

7.4 Higher dimensional transformation

The main challenge in transforming high dimensional queries is that the DPR transformation tends to retrieve a lot of false positives. We use the pruning box query to eliminate false positives and reduce the number of page accesses for execution of the query. In the following subsections, we present the results for multi-dimensional synthetic data.

7.4.1 Effect of database size

Figure 11 shows effect of database size on the query cost. We used a database with 10 dimensional vectors. Query range was kept constant. As can be seen from the figure, as the database size increases, cost for both range and box queries increases, as expected, but the rate of increase is much slower for pruning box queries than for range queries. We get about 25% reduction in the cost for a database of 9 million vectors.

7.4.2 Effect of query ranges

Figure 12 gives the comparative performance of range queries versus pruning box queries with increasing query range (chosen so as to get reasonable number of hits). As seen from the figure, performance of pruning box queries is consistently better than range queries, and the performance difference gets wider with increasing query ranges. This is because hyper-diamonds of the range queries tend to intersect more with the bounding boxes than the pruning box queries.

7.5 Performance results for real data

The experimental results described so far were carried out on synthetic data. In this section we describe effectiveness of the proposed approach on real data. We used two different datasets for our experiments.

First is a GIS dataset with two dimensions (co-ordinates of points obtained through GPS) and there are totally 108779 records (obtained from a GIS company). We randomly selected 100 points from the database as range query centers. For each query center, range was changed from 0.01 to 0.05. Figure 13 shows that even for this small database pruning box query has better performance than the traditional range query on the packed R-Tree. Our second dataset is an image feature dataset. Similar to the features used in Coral Image data [30], first three moments were calculated for hue, saturation and intensity values of the pixels. The resulting feature vector has 9 dimensions. A feature database of 10 million images obtained from [20] was built. Range was varied from 0.005 to 0.010. Note that the value of the range was chosen such that we get a reasonable number of hits. Further, the number of page accesses for 10% Linear ($\approx$ 49000) is too high and hence omitted. Figure 14 shows that with increasing range, performance improvement due to PBQ increases. which highlights applicability of the proposed transformation to high-dimensional spaces.

8 k-NN QUERIES

k-NN (k-Nearest neighbor) queries can be considered as a special case of range queries which only return the set of k records that are most similar to query record. A k-NN query can be implemented by keeping track of distance d_k of current k^{th} neighbor [12], [27]. Any data node whose MBR is farther than d_k can be safely pruned. This is conceptually similar to range query with range d_k. As query execution proceeds, the range decreases. Due to the underlying similarity between k-NN queries and range queries the proposed transformation can also be applied for k-NN queries.

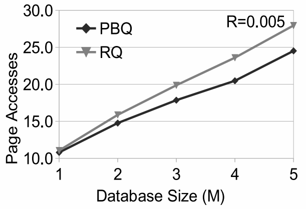


Fig. 9: Effect of database size(2-D)

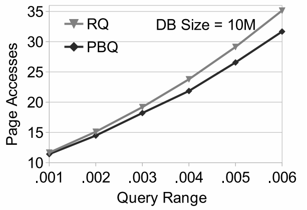


Fig. 10: Effect of query radius(2-D)

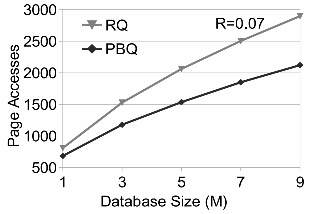


Fig. 11: Effect of database size(10-D)

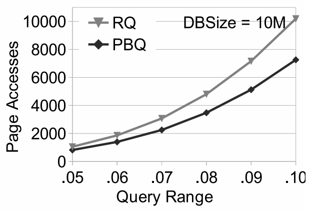


Fig. 12: Effect of query radius(10-D)

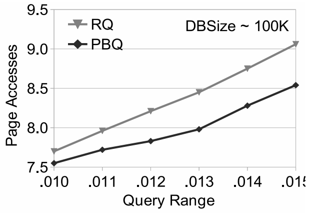


Fig. 13: Effect of query radius(GIS data)

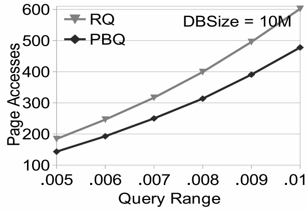


Fig. 14: Effect of query radius (image feature data)

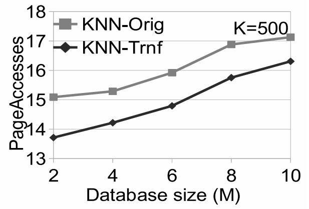


Fig. 15: KNN-Effect of database size(2-D)

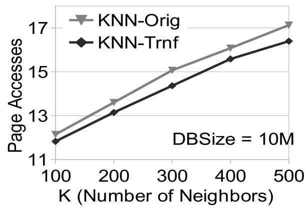


Fig. 16: KNN-Effect of increasing K(2-D)

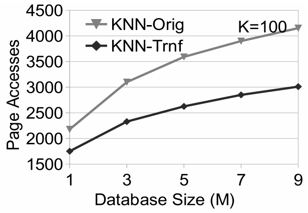


Fig. 17: KNN-Effect of database size(10-D)

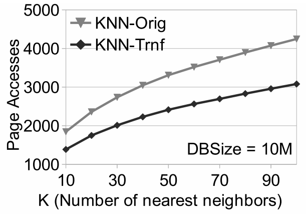


Fig. 18: KNN-Effect of increasing K(10-D)

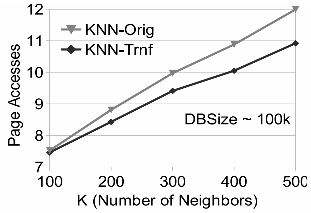


Fig. 19: KNN-Effect of increasing K(GIS data)

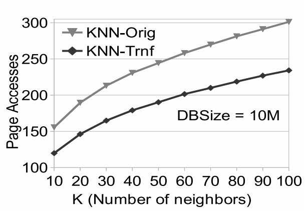


Fig. 20: KNN-increasing K(Image feature data)

Roussopoulos et. al. [27] propose some of the basic distance based heuristics for running nearest neighbor queries in R-Tree based indexes. They propose using depth-first traversal of the index tree. However, as shown in [12], best-first search is a better approach for tree traversal. Further, authors also note that not all the heuristics proposed in [27] can be extended for use in k-NN $(k > 1)$ queries. Hence, we use the best-first algorithm proposed by [12] for our experiments.

It was observed that at higher levels in the index tree, node MBRs are large in size and query point is inside a lot of MBRs and its minimum distance from these MBRs is zero. We break the ties in such cases using distance of the query point from the center of the MBR. This heuristic was observed to perform better than choosing any one of the qualifying MBRs randomly.

In the next few subsections we present results of using the transformation for k-NN queries. We executed k-NN queries in transformed space as well as original space. Different combinations of number of dimensions, database size and value of k were used. In all the following graphs, the labels KNN-Orig and KNN-Trnf are used for representing KNN queries in original space and transformed space respectively.

8.1 k-NN in 2-D Space

Figure 15 shows the results of running 500-NN queries on 2-D data. Database size was varied from 2 million to 10 million records. It can be seen from the figure that transformed space queries consistently perform better than those in the original space. Figure 16 shows the results of varying the value of k (i.e. number of neighbors) for a database of 10 million records. We observed that for very small values of $k(< 100)$, queries in transformed space perform worse than those in original space. This is expected because for small k the effective radius of the query space is much smaller. From equations 9 and 11, it can be seen that for small query radius, the transformation may not be very effective. So our observations are in accordance with the analysis. As k increases, we can see from the figure that performance improvement due to transformation increases.

8.2 k-NN in 10-D Space

Figures 17 and 18 highlight advantages of transformation for K-NN queries in 10 dimensional space. It can be clearly seen from the figures that improvement in excess of 20% can be achieved using the transformation. Further, the improvement increases with increasing database size and increasing values of k.

8.3 k-NN on Real Data

We used the proposed technique for the real database mentioned in section 7.5. Our findings are in accordance with the observations we made with synthetic data. For 2 dimensional real data, there is no improvement if the value of k is small.

But for $k \geq 80$, the transformation indeed provides better performance. For the 9-dimensional image feature data, we see about 22% improvement even for lower values of k.

9 Topological Transformation in Multi-Dimensional Space

Our PBQ approach creates bounding boxes for efficient implementation of range queries. While having the advantages in IO accesses due to their alignment with the bounding boxes of the index tree, these boxes have empty space, which may produce false positives. This extra space can be removed using the pruning box query discussed earlier. However, the amount of extra space increases as the dimensionality increases, leading to more CPU time to prune the extra space. In this section, we introduce a topological transformation algorithm to minimize the bounding box querying time for d-dimensional range queries, for $d > 2$. When $d = 3$, the algorithm for computing optimal bounding box for a 3D-cross polytope (i.e., 3D-convex polygon) exists [24]. For example, the 3D regular diamond shown in Figure 21 is in the up-right position, *i.e.*, vertices are (1,0,0), (-1,0,0), (0,1,0), (0, -1,0), (0,0,1) and (0,0,-1) We obtain the optimal bounding box for this 3D-diamond by rotating parallel to the XY-plane by 45°and then rotating clockwise by 54.7327°parallel to the YZ-plane. However, when $d > 3$, there is no such algorithms for finding optimal bounding boxes in prior literature. In this section, we present an algorithm that computes an optimized axis aligned bounding box for d-dimensional cross polytopes (*i.e.*, d-dimensional hyper convex polygon).

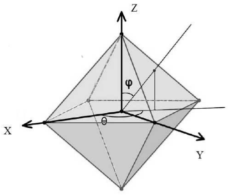


Fig. 21: 3D-diamond in upright position

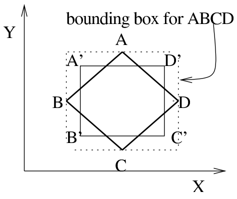


Fig. 22: Projection of 3D-diamond on XY-plane

9.1 Definitions

We first define some terminologies and prove a theorem, which will serve as the basis of our algorithm.

Definition 1: Axis Aligned Bounding Box An axis aligned bounding box of a d-dimensional polytope P is defined as the box whose body diagonal is delimited by the minimum and maximum axial coordinate values of P.

The edges of an axis aligned bounding box are parallel to the axes of the coordinate space. For example, the box A'B'C'D' in Figure 22 is the axis aligned bounding box of the diamond ABCD.

Definition 2: Oriented Bounding Box An oriented bounding box of a d-dimensional polytope P is a tightly fitting rectangular bounding box of an arbitrary orientation in the d-dimensional space.

The edges of an oriented bounding box may not be parallel to the axes.

Definition 3: Orientation in d-Dimensions Given an arbitrary polytope P in d-dimensions, all orientations of P can be obtained by rotating P in parallel with one or more of the axial planes in the d-dimensional space while the vertex coordinates of the axes that are not involved in the rotation remaining fixed.

For example, if we rotate the 3D-regular diamond of Figure 21 parallel to the XY-plane, the Z values of the vertices remain fixed.

Definition 4: Local Optimal For a given k, $0 \leq k \leq d-1$, if the axis aligned bounding box of a d-k dimensional projection of a d-dimensional polytope is its arbitrarily oriented minimum bounding box (requiring no rotation of the projection to make it smaller), then the projection is said to be local optimal.

For example, the diamond ABCD of Figure 22 is the 2D-projection of the 3D-regular diamond of Figure 21 in the XY-plane. Projection ABCD is not locally optimal because the axis aligned bounding box for this projection is A'B'C'D' as shown by the dotted line in the figure, which is not the smallest bounding box of the projection. We get the optimal axis aligned bounding box for this projection when we rotate the projection ABCD by a 45°. Note that we are not rotating the 3D diamond, instead, we are just rotating the projection if it's bounding box is not already optimal.

On the other hand, if the 3D-regular diamond itself is rotated from it's up-right position of Figure 21 by a 45°parallel to the XY-plane, the XY-projection of this newly oriented 3D-diamond is an axis aligned square which is also it's optimal axis aligned bounding box. Thus, the XY-projection of this newly oriented diamond is locally optimal. Using this definition of local optimality, we can further elaborate on the situations where multiple projections of a given orientation of a cross-polytope exhibit local optimality at the same time.

*Definition 5: Simultaneously Local Optimal (***SLO***)* For a given k, $0 \leq k \leq d-1$, if the *d-k* dimensional projections of the *d*-dimensional polytope in each d-k dimensional plane are locally optimal, then the projections are called simultaneously local optimal.

For example, when the 3D diamond of Figure 21 in upright position is first rotated parallel to the XY-plane by 45°and then rotated clockwise by 54.7327°parallel to the YZ-plane, the projections of this orientation of the diamond in all three coordinate planes are locally optimal.

If the bounding box of a d-dimensional cross polytope is minimal, then we postulate about the nature of its projections in lower dimensions in terms of SLO by the following theorem.

Theorem 9.1: Given an optimally oriented convex d-polytope P and a given k, $0 \leq k \leq d-1$, projections of the object P in $d-k$ dimensional subspaces are simultaneously local optimal.

Proof: By way of contradiction, we assume there exists a d-polytope P and rotation R such that an axis-aligned minimum bounding box B is also the minimum arbitrarily oriented minimum bounding box. Suppose that the projections of the object P, rotated by R, in all (permutations) of $d-k$

dimensions for a given k, $0 \leq k \leq d-1$, are not simultaneously local optimal. Thus, there exists a projection p' of P in $d-k$ dimensions rotated by R such that *B'* (corresponding to the projection of B along the same $d-k$ dimensions) is not an arbitrarily oriented minimum bounding box of the projection p' of P rotated by R. Then there exists an arbitrarily oriented minimum bounding box B'' of the projection p' of P rotated by R that is different from B'. Let R' be the rotation that rotates B'' such that it is an axis-aligned minimum bounding box. The volume of the bounding box is the product of the ranges in each dimension. Since projection discards dimensions not subject to the projection, any rotation of the projection does not change the range values on the axis in the orthogonal dimensions that are not subject to projection for an axis aligned bounding box. However, there does now exist a rotation of the projection such that an axis aligned bounding box is smaller than B'. Hence, the composition of R and R' is a rotation that will rotate the d-polytope P such that a smaller axis-aligned minimum bounding box exists. This contradicts our earlier assumption of the optimality of the bounding box of the d-polytope. Hence, the projections of P in $d-k$ dimensions are simultaneously locally optimal. □

The above theorem works for any d-dimensional cross polytope. In our case, the query is a d-dimensional regular hyper-diamond and if it is optimally oriented then, according to the above theorem all of it's 2-D projections are simultaneous local optimal. We use this property to develop the algorithm presented in the following section to compute optimized bounding box.

9.2 Computing Optimized Axis Aligned Bounding Box

We next introduce our algorithm called SLO for computing optimized axes aligned bounding boxes. This algorithm first picks the 2D-projection in XY-plane of the arbitrarily oriented d-dimensional regular hyper-diamond and checks if it is locally optimal. If it is not locally optimal, this projection is rotated in the XY-plane such that it becomes locally optimal. The same amount of rotation, parallel to the XY-plane, is then applied to the hyper diamond. This rotation of the hyper diamond now creates a set of new 2D projections. Now the projection in the YZ plane is rotated to make it locally optimal and then apply the same amount of rotation to the hyper diamond parallel to YZ plane. This process is repeated iteratively for all the 2D projections, wrapping around after rotating in all the planes, until all the projections are simultaneously local optimal. The projections will converge to a simultaneous local optimal in a finite number of iterations because each rotation reduces the area of projection to achieve local optimal while the lengths of the bounding box in other orthogonal planes do not change. The pseudo-code of our algorithm is shown in Algorithm 2.

Algorithm 2 Algorithm SLO

```
Input: Data points in $d$-dimensions.
  $P \leftarrow Datapoints$
  $Proj \leftarrow$ All 2D projections of $P$
  $n \leftarrow NumberofProjections$
  Initialize all $check[i]$ to 0.
  while $check[1] * check[2] * ...... * check[n] \neq 1$ do
      for $i = 1 \rightarrow n)$ do
          if $IsOptimal(Proj[i]$ then
              $check[i] \leftarrow 1$
          else
              $R \leftarrow RotatingCallipers(Proj[i])$.
              Apply $R$ to $P$.
              $Proj \leftarrow$ All 2D projections of $P$
              $check[i] \leftarrow 0$
          end if
      end for
  end while
  $FinalVolume \leftarrow$ Volume of P.
  $Orientation \leftarrow$ Optimal Rotation of P.
  return $FinalVolume$, $Orientation$
```

For example, we apply the above algorithm to the 3D-regular diamond of Figure 21 to compute the optimal orientation of the diamond corresponding to the minimum-volume axis-aligned bounding box. The algorithm starts with the diamond in upright position. It derives the three 2-D projections of the diamond, namely projections in XY, YZ and ZX planes. We number these planes as 1, 2 and 3 respectively, to correspond to the notation in the algorithm. check(1) through check(3) are all 0, initially, because the projections of the starting orientation of the diamond are assumed to be not locally optimal. For-loop starts with plane 1 (i=1 in the algorithm). $IsOptimal(Proj[1])$ checks to see if projection 1 is locally optimal, which is not the case here. Thus, the algorithm applies $RotatingCallipers(Proj[1])$ [28] next. It rotates projection 1 by 45°to make Project[1] locally optimal. Same amount of rotation ("Apply R to P" in the algorithm) is then applied to the diamond. For this newly arrived orientation of the diamond get all 2D projections again and set check[1] to 0 (not part of a simultaneous local optimal yet). Now run the for-loop for i=2 (YZ plane). $IsOptimal(Proj[2])$ checks to see if projection 2 is locally optimal, which is not the case. Thus, the algorithm applies $RotatingCallipers(Proj[2])$ next. It rotates projection 2 by 54.7327°to make it locally optimal. For this new orientation of the diamond get all 2D projections again and set check[2] to 0. The process continues for i=3 but $IsOptimal(proj[3])$ is true and check[3] is set to 1. While-loop starts again because for-loop has ended and check[1]*check[2]*check[3] is false. This time, for each value of i in the for-loop $IsOptimal(proj[i])$ is true. Therefore, check[i] is set to 1 for each i and the while loop terminates, giving the rotation of 45°parallel to XY plane and then 54.7327°parallel to YZ plane as the optimal orientation of the 3D diamond.

We have computed the volumes of optimized bounding box for d-dimensional simple hyper diamond where the diamond is centered at the origin and has a unit radius in L_1 norm. In 3D the vertices are (1,0,0), (-1,0,0), (0,1,0), (0, -1,0), (0,0,1) and (0,0,-1) Table 5 compares the volumes of the bounding boxes for d-dimensional hyper diamonds for the upright position, the method using PBQ and the method using SLO. The table shows that SLO provides 99.99% better optimized bounding box volume over the PBQ approach in 16 dimensions. Using the optimized orientation of the d-dimensional regular diamond, we have implemented range queries using this optimized box queries.

Dim	4	6	8	10	12	16
Upright	16	64	256	1024	4096	32768
PBQ	4	8	16	32	64	256
SLO	1.005	0.9473	0.3076	0.1354	0.0435	0.0022
SLO-PBQ(%)	74.88	88.16	98.08	99.58	99.93	99.99

TABLE 5: Volume Optimization for Bounding Box for Unit Hyper-Diamond

9.3 Topological Transformation based on Simultaneous Local Optimal

Based on the final orientation of the polyhedron given by the algorithm, we can obtain the transformation matrix, which then can be applied directly to transform the corresponding d-dimensional space. This transformation is achieved as follows: If A is the matrix defining the initial orientation of the d-dimensional hyper-diamond P and X is the matrix defining the final orientation of P (derived from algorithm) then we can get the topological transformation matrix T by $T = A \times X^{-1}$. Thus, a given d-dimensional query P in the original space becomes the query P' in the transformed space as $P' = P * T$.

9.4 Performance Evaluation

We conducted experiments using the same machines and the same system parameters as described in section 7. Table 6 shows the run time performance of range queries, for query ranges varying from 0.05 through 0.08, in the original space (RQ), using bounding box query of PBQ and the optimized bounding box query of SLO. Reduced run time of SLO over PBQ is due to less false positives that need to be discarded in SLO over PBQ. The number of IO's for both SLO and PBQ is the same because both approaches remove false positives before accessing the leaf nodes. Here the database size is fixed at 1 million feature vectors. We observe that the performance improvement of SLO over PBQ increases with dimensions and this improvement is about 34% for 20 dimension. The data also shows that this improvement percentage is almost the same for varying query ranges when the dimension is fixed.

Table 7 gives the performance improvement of Box queries due to SLO over that due to PBQ with increasing database size. Query range was fixed at 0.05 and the number of dimensions is 10. Performance of range query (RQ) in the original space is also included in the table for comparison purpose. We observe that the performance improvement of SLO over PBQ is stable as the database size increases.

Dim	0.05	0.06	0.07	0.08	SLO Over PBQ
8	3.215/ 3.075/ 2.733	5.235/ 3.162/ 2.82	7.761 / 6.844 / 6.072	10.969/ 9.476/ 8.481	11%
12	10.095/ 8.837/ 7.449	17.134/ 14.61 / 12.257	27.649/ 20.645 / 17.362	37.108/ 29.319/ 24.745	16%
16	41.463/ 34.89/ 26.495	63.85/ 52.469/ 39.934	92.491/ 74.474/ 56.51	123.47/ 98.109/ 74.68	24%
20	98.456/ 84.561/ 56.402	121.63/ 94.903/ 62.73	170.187/ 131.495/ 87.707	225.224/ 173.661/ 115.172	34%

TABLE 6: Run time in seconds with increasing dimensions (RQ/PBQ/SLO)

DB Size	1M	2M	5M	10M
RQ	6.478	13.301	16.273	34.274
PBQ	5.069	8.361	14.638	29.273
SLO	4.404	7.207	12.515	24.823

TABLE 7: Performance for increasing database size

10 Conclusion and Future Work

In this paper, we present two novel topological transformation schemes to improve the performance of range queries. First transformation scheme is based on a simple topological transformation and is computationally easy to implement. It provides better IO performance than that in the original space. This transformation is precise for 2 dimensions and for dimensions higher than two, additional computation is required to remove the false positives.

In the second transformation scheme amount of false positives is significantly reduced by using optimized bounding boxes. A novel approach to compute optimized bounding box in higher dimensions is given. It is shown that with increasing dimensions, the second transformation scheme requires much less computation time than that for the first.

References

[1] C. C. Aggarwal, "On the effects of dimensionality reduction on high dimensional similarity search," in *PODS '01: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. New York, NY, USA: ACM, 2001, pp. 256–266.

[2] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: an efficient and robust access method for points and rectangles," *SIGMOD Rec.*, vol. 19, no. 2, pp. 322–331, 1990.

[3] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[4] S. Berchtold, D. A. Keim, and H.-P. Kriegel, "The X-tree: An index structure for high-dimensional data," in *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996, pp. 28–39.

[5] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, no. 2, pp. 188–228, 2002.

[6] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 426–435.

[7] R. Fenk, V. Markl, and R. Bayer, "Interval processing with the ub-tree," in *Proceedings of the 2002 International Symposium on Database Engineering & Applications*, ser. IDEAS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 12–22. [Online]. Available: http://portal.acm.org/citation.cfm?id=646291.687082

[8] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Comput. Surv.*, vol. 30, pp. 170–231, June 1998. [Online]. Available: http://doi.acm.org/10.1145/280277.280279

[9] A. Guttman, "R-trees: a dynamic index structure for spatial searching," *Proceedings of ACM SIGMOD*, pp. 47–57, 1984.

[10] D. Hilbert, "Über die stetige abbildung einer linie auf ein flächenstück," vol. 38, 1890, p. 459460.

[11] K. Hinrichs, "Implementation of the grid file: design concepts and experience," *BIT*, vol. 25, pp. 569–592, December 1985. [Online]. Available: http://portal.acm.org/citation.cfm?id=5037.5039

[12] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Trans. Database Syst.*, vol. 24, pp. 265–318, June 1999. [Online]. Available: http://doi.acm.org/10.1145/320248.320255

[13] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces (survey article)," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 517–580, 2003.

[14] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems," *ACM Comput. Surv.*, vol. 40, no. 4, pp. 1–58, 2008.

[15] I. Kamel and C. Faloutsos, "On packing R-Trees," in *Proceedings of the second international conference on Information and knowledge management*, ser. CIKM '93. New York, NY, USA: ACM, 1993, pp. 490–499. [Online]. Available: http://doi.acm.org/10.1145/170088.170403

[16] N. Katayama and S. Satoh, "The SR-tree: an index structure for high-dimensional nearest neighbor queries," in *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 1997, pp. 369–380.

[17] H.-P. Kriegel and B. Seeger, "Plop-hashing: A grid file without directory," in *Data Engineering, 1988. Proceedings. Fourth International Conference on*, Feb. 1988, pp. 369 –376.

[18] A. Kumar, "G-Tree: A new data structure for organizing multidimensional data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 6, no. 2, pp. 341–347, 1994.

[19] S. Lang, *Linear Algebra*. New York: Springer-Verlag, 1987.

[20] MIT image dataset, "MIT CSAIL : Visual dictionary," 2010. [Online]. Available: {http://groups.csail.mit.edu/vision/TinyImages/}

[21] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," *Technical report*, 1966.

[22] J. Nievergelt, H. Hinterberger, and K. C. Sevcik, "The grid file: An adaptable, symmetric multikey file structure," *ACM Trans. Database Syst.*, vol. 9, pp. 38–71, March 1984. [Online]. Available: http://doi.acm.org/10.1145/348.318586

[23] J. Orenstein, "A comparison of spatial query processing techniques for native and parameter spaces," *SIGMOD Rec.*, vol. 19, pp. 343–352, May 1990. [Online]. Available: http://doi.acm.org/10.1145/93605.98743

[24] J. OŔourke, "Finding minimal enclosing boxes," *International Journal of Computer & Information Sciences*, vol. 14, no. 3, pp. 183–199, 1985. [Online]. Available: http://dx.doi.org/10.1007/BF00991005

[25] S. Pramanik, A. Watve, C. R. Meiners, and A. Liu, "Transforming range queries to equivalent box queries to optimize page access," *Proc. VLDB Endow.*, vol. 3, pp. 409–416, September 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1920841.1920895

[26] K. V. Ravi Kanth, D. Agrawal, and A. Singh, "Dimensionality reduction for similarity searching in dynamic databases," *SIGMOD Rec.*, vol. 27, no. 2, pp. 166–176, 1998.

[27] N. Roussopoulos, S. Kelley, and F. Vincent, "Nearest neighbor queries," in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '95. New York, NY, USA: ACM, 1995, pp. 71–79. [Online]. Available: http://doi.acm.org/10.1145/223784.223794

[28] G. T. Toussaint, "Solving geometric problems with the rotating calipers," in *Proc. IEEE MELECON*, vol. 83, 1983, p. A10.

[29] H. Tropf and H. Herzog, "Multidimensional range search in dynamically balanced trees," *Applied Informatics*, pp. 71–77, 1981.

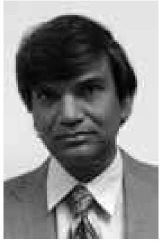
[30] UCIML Repository, "UCI machine learning repository corel image feature data set - http://archive.ics.uci.edu/ml/datasets/Corel+Image+Features," 2010. [Online]. Available: {http://archive.ics.uci.edu/ml/datasets/Corel+Image+Features}

[31] K. Vu, K. A. Hua, H. Cheng, and S.-D. Lang, "A non-linear dimensionality-reduction technique for fast similarity search in large databases," in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 2006, pp. 527–538.

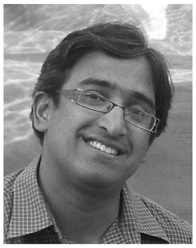
[32] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the 24rd International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 194–205.

[33] D. A. White and R. Jain, "Similarity indexing with the SS-tree," in *Proceedings of the 12th International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 516–523.

Alok Watve is currently a software engineer at Google Inc. He received his Ph.D. degree in Computer Science and Engineering from the Michigan State University in 2012 under the guidance of Dr. Pramanik . He received his MS degree from Indian Institute of Technology, Kharagpur in 2006. His research interests include database indexing, data mining and image processing.



Sakti Pramanik received the BE degree in Electrical Engineering from Calcutta University and awarded University gold medal for securing the highest grade among all branches of Engineering. He received the MS degree from the University of Alberta, Edmonton, in electrical engineering, and the PhD degree in computer science from Yale University. He is currently a professor in the Department of Computer Science and Engineering at Michigan State University.



Salman Shahid received his BE degree from the National University of Sciences & Technology, Islamabad where he is currently a research faculty. He received his Ph.D. degree in Computer Science and Engineering from the Michigan State University under the guidance of Dr. Pramanik. He was a Fulbright Scholar at the University. His research interests include computational geometry, database indexing and cyber security.

Chad R. Meiners received his Ph.D. in Computer Science at Michigan State University in 2009. He is currently full technical staff at MIT Lincoln Laboratory. His research interests include formal methods, networking, algorithms, and cyber security.

Alex X. Liu received his Ph.D. degree in computer science from the University of Texas at Austin in 2006. He is currently an assistant professor in the Department of Computer Science and Engineering at Michigan State University. He received the IEEE & IFIP William C. Carter Award in 2004 and an NSF CAREER award in 2009. He received the MSU College of Engineering Withrow Distinguished Scholar Award in 2011. His research interests focus on networking, security, and dependable systems.