A Survey on Nearest Neighbor Search Methods

Mohammad Reza Abbasifard
Faculty of Computer Engineering, Iran University of science and technology, Tehran, Iran

Bijan Ghahremani
Faculty of Computer Engineering, Iran University of science and technology, Tehran, Iran

Hassan Naderi
Faculty of Computer Engineering, Iran University of science and technology, Tehran, Iran

ABSTRACT
Nowadays, the need to techniques, approaches, and algorithms to search on data is increased due to improvements in computer science and increasing amount of information. This ever increasing information volume has led to time and computation complexity. Recently, different methods to solve such problems are proposed. Among the others, nearest neighbor search is one of the best techniques to this end which is focused by many researchers. Different techniques are used for nearest neighbor search. In addition to put an end to some complexities, variety of these techniques has made them suitable for different applications such as pattern recognition, searching in multimedia data, information retrieval, databases, data mining, and computational geometry to name but a few. In this paper, by opening a new view to this problem, a comprehensive evaluation on structures, techniques and different algorithms in this field is done and a new categorization of techniques in NNS is presented. This categorization is consists of seven groups: Weighted, Reductional, Additive, Reverse, Continuous, Principal Axis and Other techniques which are studied, evaluated and compared in this paper. Complexity of used structures, techniques and their algorithms are discussed, as well.

General Terms
Data and Information Systems, Artificial Intelligence

Keywords
Data Structure, kNN Algorithm, Nearest Neighbor Search, Query Processing

1. INTRODUCTION
By expanding computer systems, their data and information are developed and completed. Many methods with different approach are made due to this developing for searching data and finding the nearest data points. Searching the nearest neighbor in different studies are presented by different names such as post office problem, proximity search, closest point search, Best match file searching problem, index for similarity search, vector quantization encoder, the light-bulb problem and etc.[1]. The solutions for the Nearest Neighbor Search (NNS) problem usually have two parts: problem framework and their algorithms. In the framework, a formal and special explanation of the problem is created which contains object representation, distance (or similarity) function, dataset properties, dataset restrictions, computation cost model, dynamic aspects and solution requirements. In most the NNS algorithms, the main framework is based on four fundamental algorithmic ideas: Branch-and-bound, Walks, Mapping-based techniques and Epsilon nets. There are thousands of possible framework variations and any practical application can lead to its unique problem formalization such as pattern recognition, searching in multimedia data, data compression, computational statistics, information retrieval, databases and data mining, machine learning, algorithmic theory, computational geometry, recommendation systems and etc. [1-6].

A NNS problem can be defined in a metric or in a non-metric space. Metric space is defined as follow:

Definition 1. *(Metric space): Given a set S of points and d as a function to compute the distance between two points. Pair (S, d) distinguished metric space if d has characteristics such as reflexivity, non-negativity, symmetry and triangle inequality [2, 5, 7].*

Non-metric space data are indexed by special data structures in non-metric spaces and then searching is done on these indexes. A few efficient methods exist for searching in non-metric space that in most of them, non-metric space is converted to metric space. In these methods the distance formula in non-metric space is converted to a distance in metric space as this distance can be an approximate of main distance in non-metric space; but in this conversion query time is increased and accuracy is decreased[8-13]. The focus of this paper is on the problems defined on a metric space. In a more detailed classification, NNS problems can be defined in Euclidean space as follow:

Definition 2. *(Exact NNS): Given a set S of points in a d-dimensional space $R^d (S \subset R^d)$, construct a data structure which given any querypoint $q \in R^d$ finds the point in S with the smallest distance to q [2, 14].*

This definition for a small dataset with low dimension has sub linear (or even logarithmic) query time, but for massive dataset with high dimension is exponential [2]. Fortunately, approximation can decrease the exponential complexity into polynomial time. Approximate NNS is defined as:

Definition 3. *(Approximate nearest neighbor): Given a set S of Points in a d-dimensional space $R^d (S \subset R^d)$, construct a data structure which given any query point $q \in R^d$, reports any point within distance at most c times the distance from q top, where p is the point in P closest to q[2].*

Table 1. Distance Equations

No.	Distance name	Space	Equation
1	Minkowski	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = L_P[\vec{X},\vec{Y}] = \lVert x-y\rVert_p = \sqrt[P]{\sum \lvert x_i - y_i\rvert^P}$
	Manhattan	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = L_1[\vec{X},\vec{Y}] = \lVert x-y\rVert = \sum \lvert x_i - y_i\rvert$
	Euclidean	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = L_2[\vec{X},\vec{Y}] = \lVert x-y\rVert_2 = \sqrt{\sum \lvert x_i - y_i\rvert^2}$
	Chebychev	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = L_\infty[\vec{X},\vec{Y}] = \lVert x-y\rVert_\infty = \max \lvert x_i - y_i\rvert$
2	Cosine	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = \theta = \cos^{-1}\left(\frac{X.Y}{\lVert X\rVert\lVert Y\rVert}\right) = \cos^{-1}\left(\frac{\sum_{i=1}^{n} X_i \times Y_i}{\sqrt{\sum_{i=1}^{n}(X_i)^2} \times \sqrt{\sum_{i=1}^{n}(Y_i)^2}}\right) \quad 0 \leq \theta \leq 180$
3	Quadratic	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = \mathrm{dist}_M[\vec{X},\vec{Y}] = \sqrt{(\vec{x}-\vec{y})^T.M.(\vec{x}-\vec{y})}$
4	Mahalanobis	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = \sqrt{(\vec{x}-\vec{y})^T S^{-1}(\vec{x}-\vec{y})}$
5	Canberra	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = \sum_{i=1}^{n} \frac{\lvert x_i - y_i\rvert}{\lvert x_i\rvert + \lvert y_i\rvert}$
6	Energy	Vector	$\mathrm{dist}(\vec{X},\vec{Y}) = 2E\lVert x-y\rVert - E\lVert x-x'\rVert - E\lVert y-y'\rVert$
7	Edit	String	$\mathrm{dist}(S,T) = \lvert S\rvert + \lvert T\rvert - 2 \times \lvert LCS(S,T)\rvert$
8	Levenshtein	String	$\mathrm{dist}_{S,T}(\lvert S\rvert,\lvert T\rvert) = Lev_{S,T}(i,j) = \begin{cases} \max(i,j) & ,\min(i,j)=0 \\ \min\begin{cases} Lev_{S,T}(i-1,j)+1 \\ Lev_{S,T}(i,j-1)+1 \\ Lev_{S,T}(i-1,j-1)+[S_i \neq T_j] \end{cases} & ,else \end{cases}$
9	Damerau–Levenshtein	String	$\mathrm{dist}_{S,T}(\lvert S\rvert,\lvert T\rvert) = DL_{S,T}(i,j) = \begin{cases} \min\begin{cases} DL_{S,T}(i,j) \\ DL_{S,T}(i-2,j-2)+[S_i \neq T_j] \end{cases} & ,i>1\ \&\ j>1\&\ S[i]=T[j-1]\&\ S[i- \\ Lev_{S,T}(i,j) & ,else \end{cases}$
10	Hamming	String	$\mathrm{dist}(\vec{X},\vec{Y}) = \sum x_i \oplus y_i$
11	Jaccard	Set	$\mathrm{dist}(A,B) = 1 - \frac{\lvert A \cap B\rvert}{\lvert A \cup B\rvert}$
12	Hausdorff	Set	$\mathrm{dist}(A,B) = \max(\sup_{x\in A}\inf_{y\in B} d(x,y),\ \sup_{y\in B}\inf_{x\in A} d(x,y))$

The first requirement in order to search in a metric space is the existence of a formula (d) to calculate the distance between each pair of objects in S. Different metric distance functions can be defined depending on the searching space (S). Table 1 shows a list of more important formulas along with the space that they can be applied. Many researchers have presented different categorizing and evaluation of NNS [3, 4, 15, 16, 82].

Different studies have done in this field until now such as [3, 4, 82] but in this study a new categorization and aspect to this subject is presented. In the new study in present paper, different techniques and also used structures are discussed such as comparing complexities in implemented algorithms. Due to lack of complete and update categorizing, in present study a more complete and newer categorizing is shown that besides of techniques, contains structures. More over the techniques in present paper are categorized by paying attention to the type of the function. One of the other positive points in this categorizing in present paper is comparableness. In this categorizing NNS techniques have divided to seven groups Weighted, Reductional, Additive, Reverse, Continuous, Principal Axis and Other techniques.

In next section a (section 2) structure that is used in NNS techniques have studied. Because of many varieties of applications, different algorithms and structures, NNS techniques should be collect and categorize. This new categorizing is presented in section 3. Each of these groups contains correlate techniques. In section 4, a complete assessment and comparison of these techniques are presented. Table 2 has shown symbols which used in present paper.

Table 2. The symbols that is mentioned in the text

Symbol	Meaning
S	Data Set
Q	Query point
P	Data Point
dist	Distance
d	Dimension
NNS	Nearest Neighbor Search

2. NEAREST NEIGHBOR SEARCH DATA STRUCTURES

One of the main parts in NNS is data structure which used in each technique. Now there are different data structures that can use for solving this problem. By paying attention to different applications and data, each of these techniques has to use structure for maintaining, indexing points and searching. Some of these structures are techniques for NNS such as LSH, Ball-Tree, kd-Tree and etc.[2]; and the other are infrastructures in some techniques such as R-tree, R* Tree, B-Tree, X-Tree and etc. A brief overview about some of these data structures is presented as follow.

LSH (Locality Sensitive Hashing) is one of the best algorithms that can be used as a technique. The LSH algorithm is probably the one that has received most attention in practical context. Its main idea is to hash the data points using several hash function so as to ensure that, for each function the probability of collision is much higher for points which are close to each other than for those which are far apart. For finding nearest neighbor by hashing query point q saved points in the bucket that contains query point q can be retrieve. by paying attention to definition points can be closer to q with probability more than P_1 and further to q with probability less than P_2[2, 5, 17-19].

Definition 4. (Locality-Sensitive Hashing): A family H is called (r ,cr, P_1,P_2)-sensitive if for any $p, q \in R^d$:

$$\begin{cases} P_H(h(q) = h(p)) \geq P_1, & \|p-q\| \leq R \\ P_H(h(q) = h(p)) \leq P_2, & \|p-q\| \geq cR \end{cases} \quad (1)$$

In order for a LSH family to be useful, it has to satisfy $P_1 > P_2$[2].

In LSH algorithm at first preprocessing should have been done. In preprocessing all data points hash by all functions and determine their buckets. In searching step query point q is hashed and after determining buckets all of its data points retrieve as the answer [2, 5].

One of the other structures that use as a technique is kd-Tree that creates for a set with n points in d-dimensional space recursive. kd-Tree and its variance remain probably the most popular data structure used for searching in multidimensional space at least in main memory. In this structure, in each step the existence space is divided by paying attention to points dimensions.

This division is continued recursively until that in each zone just a point is remained. Finally the data structure that produced is a binary tree with n level and $\log n$ depth. For searching nearest neighbor a circle is drawn with query point q as center and $|p - q|$ as radius that p is in query point q zone. With assisting of points that are interfered with the circle, the radius and p are updated. This operation is continued until up to dating is possible and finally NN is reported [2, 20-23].

Quad-Tree and Oct-Tree act similar to kd-Tree, as Quad–Tree used in two dimensional spaces and for creating tree in it, each zone in each repetition is divided to four parts. Oct-Tree used in 3D and each zone in each repetition is divided to eight parts. Searching operation in these two structures are similar to kd-Tree [24-28].

Also we can point to Ball-Tree [2, 29, 30].A Ball-Tree is a binary tree where each node represents a set of points, called *Pts(N)*. Given a data set, the root node of a Ball-Tree represents the full set of points in the data set. A node can be either a leaf node or a non-leaf node. A leaf node explicitly contains a list of the points represented by the node. A non-leaf node has two children nodes: N.child1and N.child2, where

$$\begin{cases} pts\ (N.child1) \cap pts(N.child2) = \emptyset \\ pts\ (N.child1) \cup pts(N.child2) = pts(N) \end{cases} \quad (2)$$

Points are organized spatially. Each node has a distinguished point called a Pivot. Depending on the implementation, the Pivot may be one of the data points, or it may be the centroid of *Pts(N)*. Each node records the maximum distance of the points it owns to its pivot. Call this the radius of the node:

$$N.radius = \max_{\forall x \in N} |N.pivot - x| \quad (3)$$

Nodes lower down the tree have a smaller radius. This is achieved by insisting, at tree construction time, that

$$\begin{cases} x \in pts\ (N.child1) \rightarrow |x - N.child1.pivot| \leq |x - N.child2.pivot| \\ x \in pts\ (N.child2) \rightarrow |x - N.child2.pivot| \leq |x - N.child1.pivot| \end{cases} \quad (4)$$

Ball-Trees are constructed top down. There are several ways to construct them, and practical algorithms trade off the cost of construction against the tightness of the radius of the balls [2].

For searching in this tree, the algorithms such as KNS1, KNS2, KNS3 and KNSV can be used [2, 29]. As these algorithms have rules for pruning points.

One of the other important extant structures is R-Tree that also named *spatial access method.*

R-trees are a generalized B-tree. R-trees can handle multidimensional data. R-Trees can be used in *Temporal* and *Spatial* Data and also in commercial database management systems such as Oracle, MySQL and Postgre SQL. Furthermore in spaces which points are moveable, R-Tree is one of the most usable structures. This tree one of data structures which operate based on local indexing. This local indexing is defined as rectangular vicinity named MBR (Minimum Bounding Rectangle). MBR is the smallest local rectangle that contains its all points and subset nodes. R-Tree uses three concept distance for searching: *MinDist, MaxDist* and *MinMaxDist*. Also this tree is a balance tree and all of its leaves are in the same level. For R-Tree there are two algorithms for searching nearest neighbor to HS and RKV that HS is a *breadth search Algorithm* and RKV is a branch and bound algorithm that use *depth search* [24, 31-35].

Another structure that can be used for NNS is M-Tree that is inspirited from R-Tree and B-Tree with the difference that pay more attention to memory and I/O.

M-Tree is defined by paying attention to different situation and tries to prune more points. Searching in this is similar to R-Tree but with priority queue searching algorithm is optimum. For indexing points in metric space M-Tree is used such as VP-Trees, Cover-Trees ‹MVP-Trees and BK-Trees [36, 37].

And another structures for NNS that is created by R-Tree idea are R*-Tree, R+-Tree, TPR-Tree, X-Tree, SS-Tree, SR-Tree, A-Tree and BD-Tree [24, 31, 38-41]. At the end of this section the complexity of some of the structures are compared in table 3.

3. NEAREST NEIGHBOR SEARCH TECHNIQUE

One of the most important reasons that have made it pervasive is widespread of application and its extent. This wide spreading caused heterogeneous data, conditions and system environment and made the solution hard. So it is necessary to create a technique that has the best result. With this reason for solving NNS problem, different technique with different approach has been created. Each of these techniques can be divided to two parts. The first part consists suitable structure for indexing and maintaining data points that is discussed in the last section. It is necessary to mention that some of these structures itself can be used as a technique for NNS, such as KD-Tree, Ball-Tree and LSH.

Table 3. Computational Complexity of structures

No.	Structure's name	Idea	Construct complexity(Query Time)	Construct complexity(Space)	Search complexity(Query Time)	Search complexity (Space)
1	LSH	Mapping data point by using hash function-search based on hash function.	$O(nLkt)$	$O(nL)$	$O\left(L\left(kt + dnP_2^k\right)\right)$	$O(1)$
2	KD-Tree	Making binary tree recursively based on the mean of points- search based on nearest query region	$O(dn\log n)$	$O(dn)$	$O\left(n^{1-\frac{1}{d}} + k\right)$	$O(1)$
3	Quad-Tree	Making Quad-tree recursively based on same zones (or based on the place of points)-search based on nearest query region	$O\left((d + 1)n\log n\right)$	$O(dn)$	$O(n)$	$O(1)$
4	Oct-Tree	Making Oct-tree recursively based on axis-search based on nearest query region	$O\left((d + 1)n\log n\right)$	$O(dn)$	$O(n)$	$O(1)$
5	Ball-Tree	Making binary tree recursively based on axis-search based on data pruning and the distance from the axis.	$O(dn(\log n)^2)$	$O(dn)$	$O(dn)$	$O(1)$ $O(d\log n)$
6	R-Tree	Making R-tree into down to up based on MBR-search based on data pruning and MBR	$O(dn\log n)$	$O(dn)$	$O(dn)$	$O(\log n)$ $O(d\log_m n)$
7	M-Tree	Making M-tree into down to up based on the radius of adjacent data. Search based on data pruning and the radius of nodes.	$O(nm^2\log_m n)$	$O(n + mM)$	$O(\log_m n)$	$O(\log n)$

The second part consists a suitable algorithm for finding the nearest points to query point q. linear searching and kNN can be mentioned as simple and first techniques [2]. In linear searching for each query point q, its distance from all points in S is calculated and each point that has the lowest distance is chosen as a result. The main problem in this technique is unsalable that in high dimensional or by increasing the points in space, the speed of searching is really decreased.

kNN technique for the first time in [42] has been presented for classification and used simple algorithm. A naive solution for the NNS problem is using linear search method that computes distance from the query to every single point in the dataset and returns the k closest points. This approach is guaranteed to find the exact nearest neighbors. However, this solution can be expensive for massive datasets. By paying attention to this initial algorithm, different techniques have been presented that each of them tries to improve kNN's performance. In present study, a new, suitable and comparable categorizing from these techniques is presented. By paying attention to this, these techniques have been categorized to seven different groups that are discussed as follow.

3.1 Weighted techniques

In such these groups of techniques, by give weight to points the effect of each of them on final result is denoted that one of the main applications of this group is its usage in information classification. Some of these techniques are mentioned as follow: Weighted k-Nearest Neighbor (Weighted-kNN), Modified k-Nearest Neighbor (Modified-kNN) and Pseudo k-Nearest Neighbor (Pseudo-kNN) [43-48].

Surveying the position of each point compare to other points for query point q is one of the most application methods which Weighted-kNN use it. In this technique if distance is defined as weight, it names distance Weighted-kNN. Usually each of the points has different distance from query point q, so nearer points have more effects. In this technique calculating weight based on distance of points will be done with equation 5 [43, 44].

$$W_i = \begin{cases} \dfrac{d_k - d_i}{d_k - d_1}, & d_k \neq d_1 \\ 1, & d_k = d_1 \end{cases} \qquad (5)$$

If space has imbalance data, it is better to use Neighbor Weighted-kNN instance of Weighted-kNN. If in a set of data same of the classes has many members in compare to others, its score very high and so more query belong to this class. For solving this problem it is necessary that the classes with more members gain low weight and the classes with less members gain high weight. The weight for each class calculated from equation 6 [45, 46].

$$W_i = \frac{1}{\left(Num(C_i^d)/Min\{Num(C_l^d)|l = 1, \dots, K^*\}\right)^{1/Exponent}} \qquad (6)$$

For example in point classification for presenting better answer instead of distance, product of distance and weight must be used. Here by paying attention to space weight is calculated by one of these techniques.

In more problems choosing neighbors based on distance have some problems such as low Accuracy and incorrect answers. Modified-kNN method which use for classification, tries to solve these problems. At first a preprocessing is done on all of the points and gives a validity value to each point. This value defines based on each point H nearest neighbor which is calculated with equation 7.

$$validity(x) = \frac{1}{H}\sum_{i=1}^{H} S\left(lbl(x), lbl\left(N_i(x)\right)\right) \qquad (7)$$

And then it is continued similar to weighted-kNN with this difference that calculated weight for each point has been product in validity value and new weight is calculated. Equation 8 shows this operation [47].

$$W(i) = validity(i) \times W_i \qquad (8)$$

Pseudo-kNN technique tries to present suitable answers by increasing calculations and omitting the effects of outlier data and use *Local Mean Learning* for this. In this method it is assumed that M classes have n_i member in each. Also $x_j^{(1)}, \dots, x_j^{(k)}$ show k nearest neighbor query point q in j^{th} class with $dist_j^{(1)}, \dots, dist_j^{(k)}$ distances that is sorted ascending. For finding Pseudo-kNN, k nearest neighbor to q in each class calculated that to i^{th} near neighbor in each class weight $w_i = \frac{1}{i}$ $(i = 1, 2, \dots, k)$ is given. This weight shows the high effect of near point compare to further point. By paying attention to *Local Mean Learning,* y_j can define as follow:

$$y_j = w_1 \times dist_j^{(1)} + w_2 \times dist_j^{(2)} + \dots + w_k \times dist_j^{(k)} \; j = 1, \dots, M \qquad (9)$$

Now the class that minimize y_j is q class [48].

3.2 Reductional techniques

One of the necessary needs in data processing is extra data and its suitable summarize. This is introduced in spaces which has massive data or data that need high memory. These techniques caused improving performance of systems by decreasing data. Condensed k-Nearest Neighbor (Condensed-kNN), Reduced k-Nearest Neighbor (Reduced-kNN), Model Based k-Nearest Neighbor (ModelBased-kNN) and Clustered k-Nearest Neighbor (Clustered-kNN) are discussed in this section [49-55].

Data that are considered as unnecessary information, identical with other data, are deleted in Condensed-kNN. There are two approaches, A) It is assumed that the data are labeled. Then instead of saving the data along with their classes, sample data is saved so that there will be no duplicate data in the dataset. B) Data might be without label; thus the desired cluster can be found by clustering and sample data is obtained from the center of the cluster. kNN operation then, is carried out on the remainder of the data [49-51].

Another approach, which uses the nearest neighbor to reduce information volume, is Reduced-kNN (a developed version of condensed-kNN). In addition to removing identical data, null data is also deleted. This even shrinks more the data volume and facilitates the system response to queries. Moreover, smaller memory space is required for processing. One of the drawbacks is increase in complexity of computations and costs of execution of the algorithm consequently. In general, these two approaches are time consuming [49, 52].

The next technique to reduce information volume is ModelBased-kNN. As the technique dictates, a data model is first extracted from the information and replaces the data. This removes a great portion of the information volume. It is noticeable, however, that the data model needs to resemble well the whole data. In place of the whole data, for instance, one may use the data that show the points (usually the central point), number of the member, distance of the farthest data from the resemblance and the class label in some cases. Modelbased-kNN employs "*largest local neighborhood*" and "*largest global neighborhood*" to create a simpler structure and definition of the model so that the data are modeled step by step [53, 54].

Another technique to reduce information volume and heighten the accuracy is clustered-kNN. To reduce the data volume, clusters are utilized and the initial data are replaced by the clusters. One reason to use the technique is its higher accuracy when the clustered data and classes have outlier data. This, for a query point q, ensures that there is a point in another class closer than a point in the same class. The procedure includes removing outlier data according to a definite process known as "*austerity*" and standards (e.g. 20% of the outlier points). Then, the data are clustered by different methods such as "*k-means*" and center of the cluster are taken instead of the data. Now, class q is deterred as follows.

$$y(W_{ij}, C_j) = \begin{cases} \dfrac{Num_{ij}}{N_{C_i}}, & W_{ij} \in C_j \\ 0, & W_{ij} \notin C_j \end{cases} \qquad (10)$$

$$P(q, C_j) = \sum_{W_{ij} \in KNN} SIM(q, W_{ij}).y(W_{ij}, C_j) \qquad (11)$$

According to these equations, j^{th}class with highest value for $P(q, C_j)$ is adopted as the answer [55].

3.3 Additive Techniques

In this group of techniques it is tried to increase system operation accuracy by increasing data volume. Another aim of these techniques is paying attention to all of points together that can affect each other. Nearest Feature Line (NFL), Local Nearest Neighbor (Local-kNN), Center Based Nearest Neighbor (CenterBased-kNN) and Tunable Nearest Neighbor (Tunable-kNN) are discussed in this section [56-61].

When the number of points for classification is not enough, accuracy of the final result is unacceptable. It is necessary to have another technique for increasing data volume and the accuracy consequently. One answer, Euclidean and 2D spaces, is NFL technique. By converting each two points in a class (*future points*) into a line, not only NFL increases the data volume but it adds to effect of the points in each class. Future points (FP) symbolize features of a class and there are two FPs at least in each class. The lines drawn between the two FPs are called *future lines* (FL). Instead of calculating distance between q and other data points, perpendicular distance between q and each FL is measured (Figure 1). With high efficiency of the technique in small dataset, by increasing size of dataset the computation complexities is increased. There is a risk of wrong determination of nearest FL in NFL for distant q and FPs (Figure 2). Figure 2 shows that $\overline{x_3 x_4}$ is wrongly adopted [56, 57].

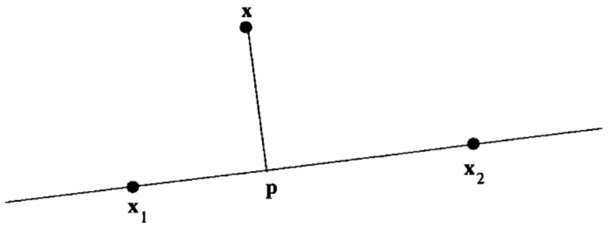

Fig 1: Mapping Query Point on Future Line [56]

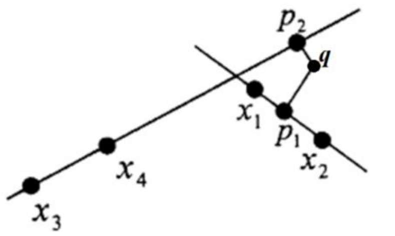

Fig 2: Nearest Future Line Drawback in Classification [58]

To deal with the drawback, Local-kNN was introduced, so that FPs are chosen among k nearest neighbors of q in each class. This ensures accurate determination of nearest FL, though great deal of computation is required. For classification, first kNN for each class$(k = 2, 3)$ is computed.

If k=2, distance between q and the FL is created from 2NN of each class and if k = 3, distance between q to future plane created from 3NN of each class are obtained. Finally, the class with shortest distance to q is taken as the result [58].

Another technique, Centerbased-kNN, tries to improve NFL. The central point of each class is computed and, then, a line is drawn from the point to the other points in the class (*center line* – CL). Afterward, nearest neighbor is computed likewise NFL so that distance between q and CL is calculated and the minimum distance is adopted [59].

When the points are distributed as figured below, there is a possibility that line between the q and the FL crosses another class – happens when neighbor data are not taken into account. Figure 3 illustrates a case when lines of two classes cross each other. This results in wrong classification and Tunable-kNN was introduced to solve the problem (also known as extended-NFL). Performance of the new method is improved by computing new distance based on the status of the data [60, 61].

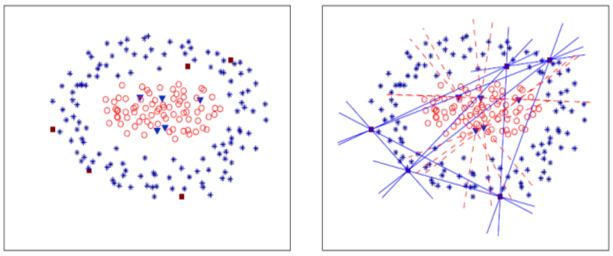

Fig 3: Lines of Two Classes Cross each other [60]

3.4 Reverse Techniques

Reverse techniques are the most important and most application techniques in NNS. This group is variety that in present paper some of them are discussed. In this group of techniques the approach of problem is changed and data points are taken more attention than query points. Reverse Nearest Neighbor (Reverse-kNN) and Mutual Nearest Neighbor (Mutual-kNN) are described in this section [62-68].

The aim of the techniques is to find data points nearest to q. Reverse-kNN uses the idea and functions according to the two definitions pointed out in equation 12.

$$\begin{array}{l} 1.\, RkNN(q) = \{r \in S \mid \forall p \in S: d(r,q) \le d(r,p)\} \\ 2.\, RkNN(q) = \{p \in S \mid q \in kNN(p)\} \end{array} \quad (12)$$

The straightest way to find reserve-kNN of query point q is to calculated the nearest point in the dataset based on the distance equation of each p; this creates regions centered by p with radius of $|p - q|$. Then, when point q is located in one of the regions, the point p in the regions is the answer. Noticeably, the for L_2-norm and L_∞-norm are circle and rectangular respectively. In spite of kNN, Reserve-kNN technique may have empty set as answer and given the distance function and dimension of the data, number of points in the set is limited. If L_2-norm is the case, for instance, we have 6 and 1_2 points at most under 3D and 2D spaces respectively. For L_∞-norm there are $3^d - 1$ equal points. A comparison between kNN and Reserve-kNN is carried out in follow [62-67].

$$\begin{cases} kNN(q) = \{r \in S \mid \forall p \in S: d(q,r) \le d(q,p)\} \\ RkNN(q) = \{r \in S \mid \forall p \in S: d(r,q) \le d(r,p)\} \end{cases} \quad (13)$$

There is a need in some applications that q is the nearest neighbor of data point in the answer set and the data point in the answer set is the nearest neighbors to query point q. In such cases, we deal with new concern known as *Mutual*-kNN that illustrated by the two definitions as follow:

$$\begin{array}{l} 1.\, MkNN_{k_1,k_2}(q) = \{p \in S \mid p \in kNN_{k_1}(q) \wedge p \in RkNN_{k_2}(q)\} \\ 2.\, MkNN_{k_1,k_2}(q) = \{p \in S \mid p \in kNN_{k_1}(q) \wedge q \in kNN_{k_2}(p)\} \end{array} \quad (14)$$

Where k_1, k_2 are number of mutual nearest neighbors. The technique adopts points that are optimum for both kNN and Reverse-kNN. In other words, following equation must be met. There are different methods to obtained Mutual-kNN *including simple processing, two-step, reuse two-heap, using NN search with pruning, and using RNN search with pruning* [68].

$$\begin{cases} MkNN_{k_1,k_2}(q) \subseteq kNN_{k_1}(q) \\ MkNN_{k_1,k_2}(q) \subseteq RkNN_{k_2}(q) \end{cases} \quad (15)$$

3.5 Continuous Techniques

Techniques that are presented in this section are suitable for points that are introduced in continuous space instead of discrete space. In this section continuous Nearest Neighbor (Continuous -kNN) and Range Nearest Neighbor (Range-kNN) are evaluated [69, 70].

The best choice to find kNN query point q on different points on a line is Continuous-kNN. For instance, for $S = \{a, b, c, d, e, f, g, h\}$ and line $\overline{se}$ continuous nearest neighbor of q on different points of the line are marked in figure 4. The continuous nearest neighbor of q is marked on different point of line. If query point q is at interval (s, s_1), then point 'a' is taken as Continuous-kNN. This for intervals (s_1, s_2), (s_2, s_3), and (s_3, e) is 'c', 'f', and 'h' respectively.

Time can be easily added as another dimension and adopt a dynamic viewpoint. That is, query point q is moving along the line. An example is the query to find "*nearest gas station on the path of point s to point e*". There are many methods to find the nearest point to *q*, and simplest cases employ an algorithm to determined points on the line called split point (a set of *split points* and start/end points for *split list*). Each point creates a region and nearest point to q is obtained when it is positioned in the regions. Another method uses R-Tree and creates a pruning algorithm to remove data point using *MinMaxDist* and *MinDist* [69].

Increase of space dimensions of q means that the point is located in multi-dimension continuous regions which raise the problem of Range-kNN. For sake of simpler NNS, the following theorem is introduced.

Theorem 1. *point p is located is Range-kNN of region Ω, if and only if p is the nearest neighbor for at least one of the margins of region Ω.*

It is noticeable that p is assumed to be outside of the region. For the 2D space it is assumed that the region is covered by four lines. Therefore, the problem is reduced to linear NNS problem. In general, for *d* dimension regions, the nearest neighbor for $2 \times d$ regions has $d - 1$ dimensions. Linear solution is adopted to deal with computation complexities [70].

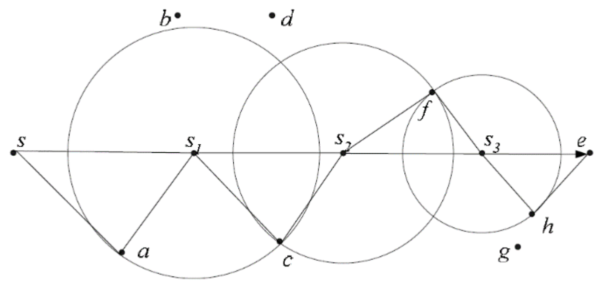

Fig 4: Continuous Nearest Neighbor Sample [69]

3.6 Principal Axis Techniques

In this group of techniques data environment is divided in several subset. Each these sets have an Axis which data are mapped on them. In this section Principal Axis Nearest Neighbor (Principal Axis-kNN) and Orthogonal Nearest Neighbor (Orthogonal-kNN) are introduced [71-74].

One of the techniques to find kNN is Principal Axis-kNN. As the method implies, the dataset is divided into several subsets. The dividing is continued until every subset is smaller than a specific threshold (e.g. 'nc'). Along with dividing, a principle axis tree (PAT) is developed so that the leaf's nodes have 'nc' data points at most (Figure 5). Each node in PAT has a principle axis which is used for mapping data and calculating distance as well as pruning. For search operation, first the node where the q is located is searched through a binary query. Then, the node and/or sibling nodes are searched to find kNN of query point q (Figure 6). To have faster process, some regions are pruned using the principle axis [71, 72].

The pruning algorithm in the technique only takes distance between q and the region axis into account (Figure 6) as position of the data point is left. Given this, a new method for pruning the data points can define. To this end, Orthogonal-kNN (also known as *Modified Principal Axis Nearest Neighbor*) is utilized. The main difference of between this and Principle Axis-kNN lies with data search and pruning algorithm. For pruning, first the chosen regions are removed and then the remaining data regions are examined for further pruning. Therefore, more points are pruned with the new function and faster is the algorithm [73, 74].

3.7 Other Techniques

In this section the techniques have been discussed that couldn't be categorized in previous groups, such as Constrained Nearest Neighbor (Constrained-kNN), Rank Nearest Neighbor (Rank-kNN), Group Nearest Neighbor (Group-kNN) and techniques that used in problems such as Road network have been introduced [75-81].

By definition NNS and majority of the available techniques are set to minimize distance between q and other points in the dataset. In some applications, there is a need to define other measures and limitations, where constrained-kNN comes handy. Two mostly used techniques to define constrained-kNN are Range Constrained Nearest Neighbor (Range Constrained-kNN) and Reverse Constrained Nearest Neighbor (Reverse Constrained-kNN) [75, 76].

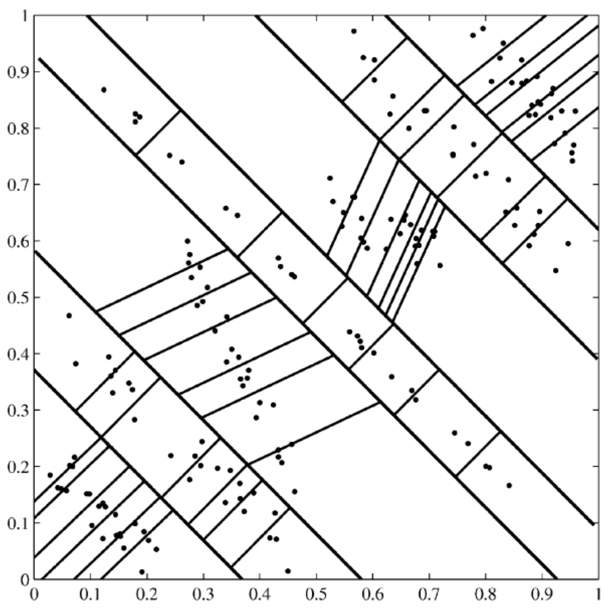

Fig 5: Partition of a 2D Data Set using a Principal Axis Tree [71]

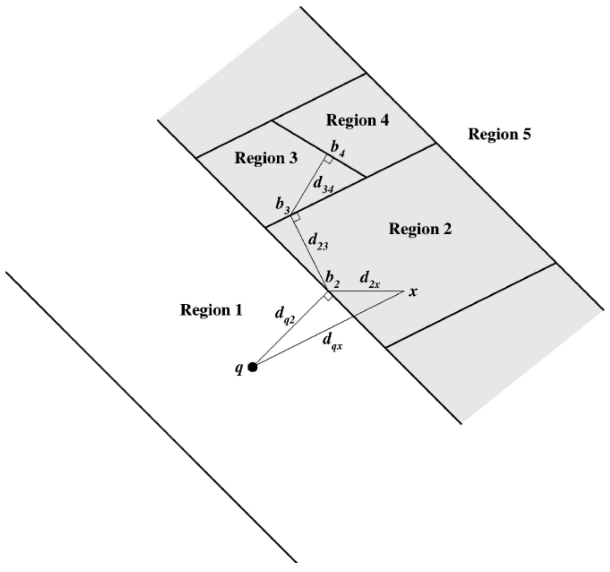

Fig 6: Pruning Data Set using a Principal Axis Tree [71]

Range Constrained-kNN: the technique is based on Euclidean space and defines conditions of location bounds. In fact, search spaces divided into specific regions. Response to query "*nearest southern point of q*", for instance, is obtained by implementing the condition of being south. Instead of a, figure 7 shows point r1 as the response to the query and the query is defined as equation 16. Other queries such "*nearest restaurant in the next town*" and "*nearest restaurant at 2km off the road*" are also possible. For the former, query of "*next town*", regions are limited to a specific region located far away from q. For the latter, query of "*road*", the regions are limited to a polygon area expanded 2km. off the road [75].

$$\{r \mid r_y < q_y \wedge \forall p \; p_y < q_y \rightarrow d(p,q) \geq d(r,q)\} \quad (16)$$

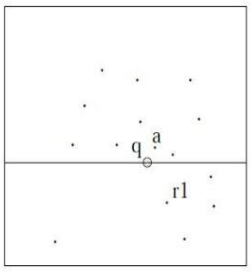

Fig 7: Constrained Nearest Neighbor Sample [75]

Reverse Constrained-kNN: the technique is based on Reverse technique and the condition is employed on the number of results. According to equation 17, when number of the results exceeds threshold of m, the same results are reported. Otherwise, the answer set is null. For the most defined spaces, number and dimension of the data points are the same. Now, Rand-kNN is a good choice, if the condition is not defined in a space (e.g. weight and length are not comparable for humans) or no data points are exist and only general information is given [76].

$$CRkNN(q) = \begin{cases} RkNN(q), & Card\big(RkNN(q)\big) \geq m \\ \emptyset, & else \end{cases} \quad (17)$$

In sum, every point in the technique (data and query) is mapped with a number according to a function $\mathrm{F}(.)$ and after sorting the results, kNN is obtained for the query point q. It is critical in Rank-kNN to find proper function $\mathrm{F}(.)$ for the data so that the output of the function must be unique for all data [77, 78].

Some of the available techniques are used for spaces where a set of query point and a set of points as answer are under consideration – nearest to all query points. This new problem is known as Group-kNN or *Aggregate Nearest Neighbor* (aggregate-kNN). "*Meet point*" problem is an example for the technique. The measure of GNN distance is defined in two ways:

1. *Sum of Distance:* where the sum of distance of data point p is obtained from the whole queries and the data with minimum sum of distance is the answer follow.

$$dist(p,Q) = \sum_{i=1}^{n} |pq_i| \quad (18)$$

2. *Maximum distance:* where distance for each data point p is calculated with each query and maximum distance is under consideration. Then the data point with minimum distance from the maximum distances is the answer follow.

$$dist(p,Q) = \underset{i=1\ldots n}{Max} |pq_i| \quad (19)$$

There is more than one method to obtain Group-kNN such as *MultipleQuery Method, Single Point Method, Minimum Bounding Method* for small number of queries, and *Group Closest Pair Method, File Multiple Query Method and File Minimum Bounding Method* for large number of queries [79, 80].

kNN technique can use in problems with graph structures, where the graph needs to meet metric space conditions. For instance, a graph of urban streets can draw (Figure 8). In such graphs, only lines (resemblance of streets) are allowed to pass. Some of the algorithms for under Aggregate-kNN for graph spaces are *Incremental Euclidean Restriction (IER), The Threshold Algorithm (TA), and Concurrent Expansion (CE)* [81].

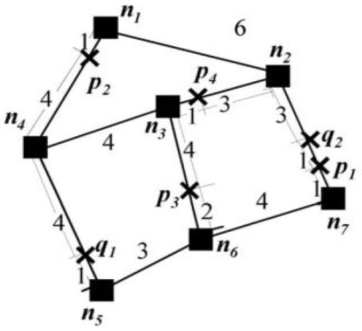

Fig 8: Graph of Urban Streets [81]

4. ASSESSMENT AND COMPARISON OF TECHNIQUES

Each of the presented techniques in this paper is suitable for using in spaces with special data but it can't be used generally. So in this section, these techniques are compared and evaluated. These comparison and evaluation are presented in to section. The first part that is shown in table 4 presented each technique's idea and applications. The second part that is shown in table 5 presented more details. In this table the different categorizes for each techniques are presented and evaluated. These categorizes cane be divided to two parts: quality and quantity categorizes. In quality categorizes it is presented that which techniques have structure and which of them don't. Also it is denoted that which techniques use all of data and which is not. In quantity categories' part the number of "*" show the size of each category. These categories contain different aspects such as *Structure Based, Whole Data, Simplicity Understanding, Difficultness Implementation, Accuracy, Volume of Data, Volume of Data, Data Dimensions, Preprocessing Cost, Search cost and Space Cost*. "-" means there is n't preprocessing in a technique and "↔" show the diversity of categories by paying attention to terms. For example in constrained-kNN search cost is depend on specify condition for NNS.

Table 4. Nearest Neighbor Techniques

No.	Technique	Idea	Application
1	Simple	Comparing query with all data points.	All
2	kNN	Searching based on almost votes.	1) Massive Data 2) Classification
3	Weighted-kNN	1) Making weight to the neighbors based on distance. 2) Making weight to the classes based on the	1) Massive Data 2) Unbalanced Data 3) Classification

		number of members.	
4	Modified-kNN	Making weight to the points based on the neighbors.	1) Classification
5	Pseudo-kNN	Making weight to the points based on their effect of dispersal.	1) Classification
6	Condensed-kNN	Omitting the repetitive and redundant data.	1) Environment with Limited Memory 2) Duplicate Data & Pattern
7	Reduced-kNN	Omitting the data which are ineffective on results.	1) Large & Null Data set
8	ModelBased-kNN	Creating model from data and replacing it instead of data.	1) Dynamic Web Mining for Large Repository
9	Clustered-kNN	Clustering data in classes and omitting far data.	1) Text Classification
10	NFL	Mapping points to lines for increasing data and accuracy.	1) Pattern Recognition 2) Face Recognition 3) Classification
11	Local-kNN	Mapping points of each group to lines distinct	1) Pattern Recognition 2) Classification
12	CenterBased-kNN	Creating lines from data points and data center.	1) Pattern Recognition 2) Classification
13	Tunable-kNN	Adjusting distance and level that are in the same level based on data condition.	1) Bias Problem 2) Classification
14	Reverse-kNN	Discussing data that are the closest to query pints.	1) Spatial Data Set 2) Business Location Planning 3) Profile Based Marketing 4) Maintaining Document Repositories
15	Mutual-kNN	Discussing data that are the closest to query pints and on the vice versa.	1) Dynamic Databases
16	Continues-kNN	Consuming query point in a straight line continuesly.	1) Nearest Data on Route 2) Mobile Data
17	Range-kNN	Consuming query point in a d-dimensional zone continusely.	1) Nearest Hotel to the City Park 2) Nearest Printer Unit to College
18	PrincipalAxis-kNN	Using the main axis to prune data.	1) Pattern Recognition 2) Spatial Data Set
19	Orthogonal-kNN	Mapping data on the main axis and prune them.	1) Pattern Recognition 2) Spatial Data Set
20	Constrained-kNN	1) The nearest neighbor in a specific zone. 2) The number of result is more than m threshold.	1) Nearest South Data 2) Nearest Restaurant in Next City 3) Client & Server Link in Busy Network
21	Rank-kNN	Ranking data and query and sorting them based on its calculated rank.	1) Multidimensional Data Points 2) Data Points with General Information
22	Group-kNN	The nearest neighbor to a group of query points.	1) Spatial Data Set 2) Meeting Point Problem

Table 5. Comparison of nearest neighbor techniques

No.	Technique	Structure Based	Whole Data	Simplicity Understanding	Difficultness Implementation	Accuracy	Volume of Data	Data Dimensions	Preprocessing Cost	Search Cost	Space Cost
1	Simple	×	√	*****	*	*****	**	**	-	*****	*
2	kNN	×	√	*****	**	**	****	***	-	*****	**
3	Weighted-kNN	×	√	****	***	***	*****	***	-	*****	**
4	Modified-kNN	×	√	****	***	****	*****	***	**	*****	***
5	Pseudo-kNN	×	√	***	***	****	*****	***	-	*****	**
6	Condensed-kNN	×	×	*****	*	***	****	↔	*****	**	***
7	Reduced-kNN	×	×	*****	*	***	****	↔	*****	**	***
8	ModelBased-kNN	×	×	*****	***	****	*****	***	*****	*	**
9	Clustered-kNN	×	×	***	****	****	*****	***	*****	*	**
10	NFL	√	√	****	**	***	*	*	*****	****	****
11	Local-kNN	√	√	****	**	****	*	*	****	***	****
12	CenterBased-kNN	√	√	****	**	****	*	*	*****	****	****

13	Tunable-kNN	√	√	**	**	*****	*	*	*****	****	****
14	Reverse-kNN	√	√	***	**	*****	**	*	*****	****	***
15	Mutual-kNN	√	√	***	***	****	**	*	*****	****	***
16	Continues-kNN	×	√	****	***	*****	*****	*	-	*****	**
17	Range-kNN	×	√	***	****	****	*****	****	-	*****	**
18	PrincipalAxis-kNN	√	√	***	****	****	*****	*	*****	*	***
18	Orthogonal-kNN	√	√	***	****	****	*****	*	****	**	***
20	Constrained-kNN	×	×	****	↔	*****	**	*	↔	↔	↔
21	Rank-kNN	√	√	***	***	****	****	*****	****	**	***
22	Group-kNN	√	√	****	****	*****	↔	**	-	****	***

5. CONCLUSION

In this paper, the metric and non-metric spaces are defined. The first one is used in NNS problem. Diverse distance formulas which are used in this problem to find the nearest data point are described and then different structures are introduced. These structures are used for indexing points and making the searching operation faster. Some of these structures such as: Ball-Tree, LSH and KD-Tree are considered as technique for NNS problem. Finally, a new categorization based on the functionality of different techniques for NNS problem is introduced. Techniques with similar functionality are grouped together in this categorization. This categorization consists of seven groups; Weighted, Reductional, Additive, Reverse, Continuous, Principal Axis and Other techniques. In each group, the main features of the group are described and each technique is introduced briefly. Finally, a complete comparison of these techniques is done.

6. REFERENCES

[1] A. Andoni, Nearest Neighbor Search - the Old, the New, and the Impossible, Ph.D. dissertation, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2009.

[2] G. Shakhnarovich, T. Darrell, and P. Indyk, Nearest Neighbor Methods in Learning and Vision : Theory and Practice, March 2006 [Online].

[3] N. Bhatia and V. Ashev, Survey of Nearest Neighbor Techniques, International Journal of Computer Science and Information Security, 8(2), 2010, pp. 1-4.

[4] S. Dhanabal and S. Chandramathi, A Review of various k-Nearest Neighbor Query Processing Techniques, Computer Applications. 31(7), 2011, pp. 14-22.

[5] A. Rajaraman and J. D. Ullman. Mining of Massive Datasets, December 2011 [Online].

[6] D. Marshal, Nearest Neighbour Searching in High Dimensional Metric Space, M.Sc. thesis, Computer Science, Australian National University, 2006.

[7] P. Zezula, G. Amato, V. Dohnal, and M. Batko, Similarity Search: The Metric Space Approach, December 2005 [Online].

[8] T. Skopal and J. Lokoc, NM-Tree: Flexible Approximate Similarity Search in Metric and Non metric Spaces, in 19th International Conference on Database and Expert Systems Applications, Turin, Italy, 2008, pp. 312-325.

[9] B. Zhang and S. N. Srihari, A Fast Algorithm for Finding k-Nearest Neighbors with Non-Metric Dissimilarity, in 8th International Workshop on Frontiers in Handwriting Recognition, Ontario, Canada, 2002, pp. 13-18.

[10] J. Lokoc, On M-tree Variants in Metric and Non-metric Spaces, in 17th Annual Conference of Doctoral Students, Prague, Czech Republic, 2008, pp. 230-234.

[11] T. Skopal, On Fast Non-Metric Similarity Search by Metric Access Methods, in 10th International Conference on Advances in Database Technology, Munich, Germany, 2006, pp. 718-736.

[12] T. Skopal, Unified Framework for Fast Exact and Approximate Search in Dissimilarity Spaces, ACM Transactions on Database Systems. 32(4), 2007,pp. 29.

[13] Y. Rubner, J. Puzicha, C. Tomasi, and J. M. Buhmann, Empirical Evaluation of Dissimilarity Measures for Color and Texture, Computer Vision and Image Understanding. 84(1), 2001, pp. 25-43.

[14] K. Fukunaga and P. M. Narendra, A Branch and Bound Algorithm for Computing k-Nearest Neighbors, IEEE Transactions on Computer. 24(7), 1975, pp. 750-753.

[15] P. N. Yianilos, Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces, in 4th Annual ACM-SIAM Symposium on Discrete Algorithms, Austin, Texas, USA, 1993, pp. 311-321.

[16] F. Bajramovic, F. Mattern, N. Butko, and J. Denzler, A Comparison of Nearest Neighbor Search Algorithms for Generic Object Recognition, in 8th International Conference on Advanced Concepts for Intelligent Vision Systems, Antwerp, Belgium, 2006, pp. 1186-1197.

[17] M. Slaney and M. Casey, Locality Sensitive Hashing for Finding Nearest Neighbors, IEEE Signal Processing Magazine. 25(2), 2008, pp. 128-131.

[18] A. Andoni and P. Indyk, Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimension, in 47th Annual IEEE Symposium on Foundations of Computer Science, Berkeley, California, USA, 2006, pp. 459-468.

[19] A. Gionis, P. Indyk, and R. Motwani, Similarity Search in High Dimensions via Hashing, in 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, 1999, pp. 518-529.

[20] R. F. Sproull, Refinements to Nearest-Neighbor Searching in k Dimensional Trees, Algorithmica. 6(4), 1991, pp. 579-589.

[21] J. L. Bentley, Multidimensional binary search trees Used for Associative Searching, Communications of the ACM. 18(9), 1975, pp. 509-517.

[22] R. Panigrahy, An Improved Algorithm Finding Nearest Neighbor Using Kd-Trees, in 8th Latin American conference on Theoretical informatics, Bzios, Brazil, 2008, pp. 387-398.

[23] A. Nuchter, K. Lingemann, and J. Hertzberg, Cached KD Tree Search for ICP Algorithms, in 6th International Conference on 3-D Digital Imaging and Modeling, Montreal, Quebec, Canada, 2007, pp. 419-426.

[24] V. Gaede and O. Gunther, Multidimensional access methods, ACM Computing Surveys. 30(2), 1998, pp. 170-231.

[25] H. Samet, The Quadtree and Related Hierarchical Data Structures, ACM Computing Surveys. 16(2), 1984, pp. 187-260.

[26] J. Tayeb, O. Ulusoy, and O. Wolfson, A Quadtree Based Dynamic Attribute Indexing Method, The Computer Journal. 41(3), 1998,pp. 185-200.

[27] C. A. Shaffer and H. Samet, Optimal Quadtree Construction Algorithms, Computer Vision, Graphics, and Image Processing. 37(3), 1987, pp. 402-419.

[28] D. Meagher, Geometric Modeling Using Octree Encoding, Computer Graphics and Image Processing. 19(2), 1982, pp. 129-147.

[29] T. Liu, A. W. Moore, and A. Gray, New Algorithms for Efficient High Dimensional Nonparametric Classification, The Journal of Machine Learning Research. 7(1), 2006, pp. 1135–1158.

[30] S. M. Omohundro, Five Balltree Construction Algorithms, International Computer Science Institute, Berkeley, California, USA, Tech, 1989.

[31] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis, R-Trees: Theory and Applications, November 2005 [Online].

[32] A. Guttman, R-Trees: A dynamic index structure for spatial searching, ACM SIGMOD Record. 14(2), 1984, pp. 47-57.

[33] M.k. Wu, Evaluation of R-trees for Nearest Neighbor Search, M.Sc. thesis, Computer Science, University of Houston, 2006.

[34] N. Roussopoulos, S. Kelley, and F. Vincent, Nearest Neighbor Queries, ACM SIGMOD Record. 24(2), 1995, pp. 71-79.

[35] M. Adler and B. Heeringa, Search Space Reductions for Nearest Neighbor Queries, in 5th international conference on Theory and applications of models of computation, Xian, China, 2008, pp. 554-568.

[36] P. Ciaccia, M. Patella, and P. Zezula, M-tree: An Efficient Access Method for Similarity Search in Metric Spaces, in 23rd Very Large Data Bases Conference, Athens, Greece, 1997.

[37] T. Skopal, Pivoting M-tree: A Metric Access Method for Efficient Similarity Search, in Dateso 2004 Annual International Workshop on Databases, Texts, Specifications and Objects, Desna, Czech Republic, 2004, pp. 27-37.

[38] S. Berchtold, D. A. Keim, and H. P. Kriegei, The X Tree: An Index Structure for High Dimensional Data, in 22th

International Conference on Very Large Data Bases, San Francisco, California, USA, 1996, pp. 28-39.

[39] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima, The A Tree: An Index Structure for High Dimensional Spaces Using Relative Approximation, in 26th International Conference on Very Large Data Bases, Cairo, Egypt, 2000, pp. 516-526.

[40] D. A. White and R. Jain, Similarity Indexing with the SS Tree, in 12th International Conference on Data Engineering, New Orleans, Los Angeles, USA, 1996, pp. 516-523.

[41] N. Katayama and S. Satoh, The SR Tree: An Index Structure for High Dimensional Nearest Neighbor Queries, ACM SIGMOD Record. 26(2), 1997, pp. 369-380.

[42] T. M. Cover and P. E. Hart, Nearest Neighbor Pattern Classification, IEEE Transactions on Information Theory. 13(1), 1967, pp. 21-27.

[43] S. A. Dudani, The Distance-Weighted k-Nearest-Neighbor Rule, IEEE Transactions on Systems, Man and Cybernetics. 6(4), 1976, pp. 325-327.

[44] T. Bailey and A. K. Jain, A Note on Distance-Weighted k-Nearest Neighbor Rules, IEEE Transactions on Systems, Man and Cybernetics. 8(4), 1978, pp. 311-313.

[45] S. Tan, Neighbor-weighted K-nearest neighbor for unbalanced text corpus, Expert Systems with Applications. 28(4), 2005, pp. 667-671.

[46] K. Hechenbichler and K. Schliep, Weighted k-Nearest-Neighbor Techniques and Ordinal Classification, Collaborative Research Center, LMU University, Munich, Germany, Tech, 2006.

[47] H. Parvin, H. Alizadeh, and B. Minaei-Bidgoli, MKNN: Modified K-Nearest Neighbor, in World Congress on Engineering and Computer Science, San Francisco, California, USA, 2008, pp. 831-834.

[48] Y. Zeng, Y. Yang, and L. Zhao, Pseudo Nearest Neighbor Rule for Pattern Classification, Expert Systems with Applications. 36(2), 2009, pp. 3587-3595.

[49] V. J. d. A. e. S. Lobo, Ship Noise Classification: A Contribution to Prototype Based Classifier Design, Ph.D. dissertation, College of Science and Technology, New University of Lisbon, 2002.

[50] P. E. Hart, The Condensed Nearest Neighbor Rule, IEEE Transactions on Information Theory. 14(3), 1968, pp. 515-516.

[51] F. Angiulli, Fast Condensed Nearest Neighbor Rule, in 22nd International Conference on Machine Learning, Bonn, Germany, 2005, pp. 25-32.

[52] G. W. Gates, The Reduced Nearest Neighbor Rule, IEEE Transactions on Information Theory. 18(3), 1972, pp. 431-433.

[53] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, KNN Model-Based Approach in Classification, in OTM Confederated International Conferences on the Move to Meaningful Internet Systems, Catania, Sicily, Italy, 2003, pp. 986-996.

[54] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, An KNN Model-Based Approach and its Application in Classification in Text Categorization, in 5th International Conference on Computational Linguistics and Intelligent Text Processing, Seoul, Korea, 2004, pp. 559-570.

[55] Z. Yong, L. Youwen, and X. Shixiong, An Improved KNN Text Classification Algorithm Based on Clustering, Journal of Computers. 4(3), 2009, pp. 230-237.

[56] S. Z. Li and J. Lu, Face Recognition Using the Nearest Feature Line Method, IEEE Transactions on Neural Networks. 10(2), 1999, pp. 439-443.

[57] S. Z. Li, K. L. Chan, and C. Wang, Performance Evaluation of the Nearest Feature Line Method in Image Classification and Retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(11), 2000, pp. 1335-1339.

[58] W. Zheng, L. Zhao, and C. Zou, Locally nearest neighbor classifiers for pattern classification, Pattern Recognition. 37(6), 2004, pp. 1307-1309.

[59] Q. B. Gao and Z. Z. Wang, Center Based Nearest Neighbor Classifier, Pattern Recognition. 40(1), 2007, pp. 346-349.

[60] Y. Zhou, C. Zhang, and J. Wang, Tunable Nearest Neighbor Classifier, in 26th DAGM Symposium on Artificial Intelligence, Tubingen, Germany, 2004, pp. 204-211.

[61] Y. Zhou, C. Zhang, and J. Wang, Extended Nearest Feature Line Classifier, in 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, 2004, pp. 183-190.

[62] F. Korn and S. M. ukrishnan, Influence Sets Based on Reverse Nearest Neighbor Queries, ACM SIGMOD Record. 29(2), 2000, pp. 201-212.

[63] C. Yang and K. I. Lin, An Index Structure for Efficient Reverse Nearest Neighbor Queries, in 17th International Conference on Data Engineering, Heidelberg, Germany, 2001, pp. 485-492.

[64] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis, Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects, The VLDB Journal. 15(3), 2006, pp. 229-249.

[65] Y. Tao, M. L. Yiu, and N. Mamoulis, Reverse Nearest Neighbor Search in Metric Spaces, IEEE Transactions on Knowledge and Data Engineering. 18(9), 2006, pp. 1239-1252.

[66] I. Stanoi, D. Agrawal, and A. E. Abbadi, Reverse Nearest Neighbor Queries for Dynamic Databases, in ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Dallas, Texas, USA, 2000, pp. 44-53.

[67] E. Achtert, C. Bohm, P. Kroger, P. Kunath, M. Renz, and A. Pryakhin, Efficient Reverse k-Nearest Neighbor Search in Arbitrary Metric Spaces, in ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, 2006, pp. 515-526.

[68] Y. Gao, B. Zheng, G. Chen, and Q. Li, On Efficient Mutual Nearest Neighbor Query Processing in Spatial Databases, Data & Knowledge Engineering. 68(8), 2009, pp. 705-727.

[69] Y. Tao, D. Papadias, and Q. Shen, Continuous Nearest Neighbor Search, in 28th international conference on Very Large Data Bases, Hong Kong, China, 2002, pp. 287-298.

[70] H. Hu and D. L. Lee, Range Nearest-Neighbor Query, IEEE Transactions on Knowledge and Data Engineering. 18(1), 2006, pp. 78-91.

[71] J. McNames, A Fast Nearest-Neighbor Algorithm Based on a Principal Axis Search Tree, IEEE Transactions on Pattern Analysis and Machine Intelligence. 23(9), 2001, pp. 964-976.

[72] B. Wang and J. Q. Gan, Integration of Projected Clusters and Principal Axis Trees for High-Dimensional Data Indexing and Query, in 5th International Conference on Intelligent Data Engineering and Automated Learning, Exeter, UK, 2004, 2001, pp. 191–196.

[73] Y. C. Liaw, M. L. Leou, and C. M. Wu, Fast Exact k Nearest Neighbors Search Using an Orthogonal Search Tree, Pattern Recognition. 43(6), 2010, pp. 2351-2358.

[74] Y. C. Liaw, C. M. Wu, and M. L. Leou, Fast k Nearest Neighbors Search Using Modified Principal Axis Search Tree, Digital Signal Processing. 20(5), 2010 pp. 1494-1501.

[75] H. Ferhatosmanoglu, I. Stanoi, D. Agrawal, and A. E. Abbadi, Constrained Nearest Neighbor Queries, in 7th International Symposium on Advances in Spatial and Temporal Databases, Redondo Beach, California, USA, 2001, pp. 257-276.

[76] T. Emrich, H. P. Kriegel, P. Kröger, M. Renz, and A. Züfle, Constrained Reverse Nearest Neighbor Search on Mobile Objects, in 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, Washington, USA, 2009, pp. 197-206.

[77] S. C. Bagui and N. R. Pal, A Multistage Generalization of the Rank Nearest Neighbor Classification Rule, Pattern Recognition Letters. 16(6), 1995, pp. 601-614.

[78] S. C. Bagui, S. Bagui, K. Pal, and N. R. Pal, Breast Cancer Detection using Rank Nearest Neighbor Classification Rules, Pattern Recognition. 36(1), 2003, pp. 25-34.

[79] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis, Group Nearest Neighbor Queries, in 20th International Conference on Data Engineering, Massachusetts, Boston, USA, 2004, pp. 301-312.

[80] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, Aggregate Nearest Neighbor Queries in Spatial Databases, ACM Transactions on Database Systems. 30(2), 2005, pp. 529-576.

[81] M. L. Yiu, N. Mamoulis, and D. Papadias, Aggregate Nearest Neighbor Queries in Road Networks, IEEE Transactions on Knowledge and Data Engineering. 17(6), 2005, pp. 820-833.

[82] H. Samet, Foundations of Multidimensional and Metric Data Structures, University of Maryland at College Park, Morgan Kaufmann, August 2006.