باشترکردنی ئەدای بنکەدراوەی فەزایی لەسەر بنەمای تێکەڵی فێربوونی ئامێری و پێکهاتەی داتای هیلی هیلبێرت

Improving Performance of Spatial Database Based on Hybrid Machine Learning and Hilbert Curve Data Structure

تحسين أداء قاعدة البيانات المكانية بناءً على التعلم الآلي الهجين وهيكل بيانات منحنى هيلبرت

گەلاوێژ محمد نجیب[1]، نزارعەبدولقادر عەلی[2]
[1] بەشی تەکنەلۆجیای زانیاری، کۆلیجی تەکنیکی ئینفورماتیك، زانکوی پولیتەکنیکی سلیمانی، شاری سلێمانی، هەرێمی کوردوستان، عێراق
[2] بەشی ئامار و ئینفورماتیک، سکولی کارگێڕی و ئابووری، زانکوی سلێمانی، شاری سلێمانی، هەرێمی کوردستان، عێراق
Corresponding author's e-mail: galawizh.najeeb@spu.edu.iq

پوختە

ئەم کارە ڕێبازێکی نوێ بۆ ئیندێکسی فرەڕەهەند دەناسێنێت و شیکاری دەکات. لەسەر بنەمای چەمکەکانی ئیندێکسکردنی فەزایی فێربووی تێکەڵە بە بەکارهێنانی ئەلگۆریتمەکانی پڕکردنەوەی بۆشایی هیلبێرت لەگەڵ فێربوونی ئامێر. بەکارهێنانی ئەلگۆریتمەکەی هیلبێرت بۆ بەدەستهێنانی ئیندێکسکردن بۆ هەر شتێکی فەزایی (خاڵ، هێڵ، فرەگۆشە)، پاشان جێبەجێکردنی پرسیارەکانی نزیکترین دراوسێ بە تەکنیکێکی تەقلیدی. بە وەرگرتنی سوود لە شێوازی فێربوونی ئامێر بۆ فێربوونی پێوەرەکانی شتە فەزاییەکان، لە شێوازی فێربوودا ئێمە ئەلگۆریتمی هیلبێرتمان بەکارهێنا بۆ ئیندێکسکردنی شتە فەزاییەکان وەک لە شێوازی تەقلیدیدا، و فێربوونی ئەو پێوەرانە، پاشان پرسیاری نزیکترین دراوسێ وەک لە تەقلیدیدا جێبەجێ بکەین، لە کۆتاییدا کاتی جێبەجێکردن حیساب دەکەین. ئەنجامێکی گرنگ کە لە ئەلگۆریتمێکی پێشنیارکراوی ئیندێکسی فێربوونی تێکەڵاو (HLI) تێدەپەڕێت کە باشتربوونی ئەدای کارکردنە بەسەر کێوی هیلبێرتدا لە شێوازی فێربوودا گەورەیە بە بەراوردکردنی نێوان شێوازە تەقلیدی و فێربووەکان کە لەڕێگەی حیسابکردنی کاتی جێبەجێکردنی هەر تەکنیکێکی پرۆسێسی پرسیار بۆ ئەنجام دەدرێت هەر سێ جۆری شتە فەزاییەکان. ئێمە هەردوو شێوازی ئیندێکسکردنمان تاقیکردەوە بۆ بەراوردکردن و هەڵسەنگاندنی هەردوو تەکنیکەکە، HLI پێشنیارکراوی ئێمە، ئەنجامە بەرچاوەکانی هەیە لەڕووی کەمتر لە کاتی جێبەجێکردنی پرسیار کە بەهۆی بەرزکردنەوەی ئەدای بنکەدراوەی فەزایی. ئیندێکسی پێشنیارکراوی کە لە ڕێگەی کێوی تایبەتمەندی کارکردنی وەرگرەوە هەڵسەنگێنداوە (هێڵی ROC-curve) بۆ مۆدێلی باشی سیستەم، هەروەها پێوەرە ئاماریەکانی MSE و R2.

وشەی سەرەکی: ئیندێکسی فەزایی، هێڵی هیلبێرت، فێربوونی ئامێر، نەریتی، ئەدای کارکردن.

گۆڤاری زانکۆی هەڵەبجە: گۆڤارێکی زانستی ئەکادیمیە زانکۆی هەڵەبجە دەری دەکات	
DOI Link	http://doi.org/10.32410/huj-10505
ڕێککەوتەکان	ڕێکەوتی وەرگرتن: ٢٠٢٣/١/١٤ | ڕێککەوتی پەسەندکردن: ٢٠٢٣/٢/١٢ | ڕێککەوتی بڵاوکردنەوە: ٢٠٢٣/١٢/٣١
ئیمەیلی توێژەر	galawizh.najeeb@spu.edu.iq

الملخص

يقدم هذا العمل ويحلل نهجًا جديدًا للفهرسة متعددة الأبعاد. يعتمد على مفاهيم الفهرسة المكانية المختلطة المكتسبة باستخدام خوارزمية منحنى هيلبرت لملء الفراغ مع التعلم الآلي. استخدام خوارزمية Hilbert للحصول على فهرسة لكل كائن مكاني (نقطة ، خط ، مضلع) ، ثم تنفيذ أقرب استعلامات جار في التقنية التقليدية. الاستفادة من طريقة التعلم الآلي لتعلم مؤشرات الكائنات المكانية ، في الطريقة التي تم تعلمها ، استخدمنا أيضًا منحنى هيلبرت لفهرسة الكائنات المكانية كما في الطريقة التقليدية ، وتعلم المؤشرات ، ثم تنفيذ استعلام الجار الأقرب كما هو الحال في الطريقة التقليدية ، وحساب وقت التنفيذ أخيرًا. النتيجة المهمة التي تتجاوز خوارزمية فهرسة التعلم المختلط المقترحة (HLI) وهي تحسين الأداء على منحنى هيلبرت رائعة في الطريقة المكتسبة من خلال المقارنة بين الطرق التقليدية والمتعلمة والتي تتم عن طريق حساب وقت تنفيذ كل تقنيات معالجة الاستعلام لـ جميع أنواع الكائنات المكانية الثلاثة. لقد اختبرنا كلتا طريقتين الفهرسة لمقارنة وتقييم كلتا الطريقتين ، HLI المقترحة لدينا ، لها نتائج مهمة من حيث وقت تنفيذ الاستعلام أقل والذي يرجع إلى تحسين أداء قاعدة البيانات المكانية. تم تقييم الفهرسة المقترحة من خلال منحنى خصائص تشغيل المستقبِل (منحنى ROC) لنموذج أمثلية النظام ، وكذلك المقاييس الإحصائية MSE و R2.

الكلمات المفتاحية : الفهرس المكاني ، منحنى هيلبرت ، التعلم الآلي، أداء، تقليدي .

Abstract

This work introduces and analyzes a novel approach to multi-dimensional indexing. It is based on the concepts of the hybrid learnt spatial indexing by using Hilbert space filling curve algorithm with machine learning. Using Hilbert algorithm to obtain indexing for each spatial objects (point, line, polygon), then executing nearest neighbor queries in traditional technique. Taking benefits of machine learning method to learn indices of spatial objects, in learnt method we also used Hilbert curve to indexing spatial objects as in traditional method, and learning those indices, then implement nearest neighbor query as in traditional, finally calculate execution time. An important result that goes beyond proposed hybrid learning indexing algorithm (HLI) that is the performance improvement over the Hilbert curve is great in learnt method by making comparison between traditional and learned methods which is done through calculating execution time of each techniques of query processing for all three spatial objects types. We tested both indexing methods to compare and evaluate both techniques, our proposed HLI, has significant results in term of less query execution time which is due to enhance performance of spatial database. Proposed indexing evaluated through receiver operating characteristic curve (ROC- curve) for system optimality model, also MSE and R2 statistical measures.

1.Introduction

Spatial database systems support spatial data types and, at the absolute least, use spatial indexing techniques in their data format and query language. Geospatial database systems, which also include spatial indexing and efficient algorithms, are the foundation upon which geographic information systems and other applications are created (Güting, 1994). The term "spatial data" refers to items with a spatial component, such as points, lines, regions, rectangles, surfaces, volumes, and even data with a higher dimension, such as time. Cities, rivers, highways, counties, and states are examples of spatial data. A spatial access strategy must take into account both spatial indexing and clustering techniques. Without a geographic index, each database item must be checked to see if it complies with the geographical selection criteria; in relational databases, this is known as a "full table scan" (Jia et al., 2022). Space-filling curves (SFC) offer a natural mapping from a high-dimensional space to a one-dimensional curve, and the arrangement of the points on SFC has been frequently used as a meaningful point arrangement (Liao et al., 2001). Dynamically formed spatial connections are produced throughout the query processing process. In order to locate spatial items efficiently using proximity, a spatial index must be created. Effective spatial operations like locating an object's neighbors and locating objects inside a certain query location must be made possible by the underlying data structure. The bulk of indexes are built on the divide and conquer principle. With this approach, hierarchical indexing structures are produced. The technique necessitates reducing searches so that fewer items are searched the more information has to be evaluated, making it a perfect fit for database systems with memory space restrictions. The advantage of hierarchical structures is that they are good range searchers. Indexing in an SDB is different from indexing in a typical database since the data in an SDB are multi-dimensional objects connected to geographical coordinates. Additionally, it offers a one by one (1:1) mapping between one-dimensional space and multidimensional space. The primary goal of spatial indexing is to improve spatial selection, or to find things from a huge collection of spatial objects that are specifically related to a given query SDT value(Ooi et al., 1993).

Related work

The neighboring blocks in the two-dimensional space must always match the adjacent line intervals in the curve in order for the Hilbert curve to have a total ordering (Davitkova et al., 2020; Kraska et al., 2018). Chen and Chang proposed the neighbor finding strategy based on the Hilbert curve (denoted as the CCSF strategy (Kraska et al., 2021). Chen and Chang proposed the neighbor finding strategy based on the Hilbert curve (denoted as the CCSF strategy(Chen & Patel, 2006). Research(Liu & Schrack, 1996) Theoretically analyze the conditions under which learned indexes outperform traditional index structures, describe the main challenges in designing learned index structures, and investigate the extent to which learned models, such as neural net-

works, can be used to augment or even replace traditional index structures such as B-Trees and Bloom filters. For processing point, KNN, and range queries, the ML-Index is a memory-efficient Multidimensional Learned (ML) structure. The ML-Index splits and converts data into one-dimensional values related to the distance to their nearest reference point shown in (Chen & Chang, 2011) using data-dependent reference points. By sequentially dividing points along a succession of dimensions into equal-sized cells and sorting them by the cell that they inhabit, a learning database system called SageDB (Chen & Chang, 2011) extends the ideas to multidimensional data. Presorting the data by their Z-order (Markl, 2000), Hilbert order(Lawder & King, 2001), or the relative distance to reference points (Jagadish et al., 2005) has been intensively investigated as a means of providing an affordable representation of multidimensional points that may be usefully sorted. For quickly addressing spatially questions, a learnt Z-order Model (Wang et al., 2019) focuses on integrating Z-order scaling with a staged learning model.

1.1.Techniques for Indexing in Spatial Database

Because of the enormous number of spatial databases, spatial access techniques are commonly employed to organize and speed up spatial item retrieval (Manolopoulos, Theodoridis, & Tsotras, 2009). A spatial index is a data structure that allows users to quickly retrieve spatial data. There are several types of spatial data, the most popular of which being points, lines, and regions (Samet, 1990). A database system needs an index mechanism to enable it retrieve data items rapidly according to their spatial positions in order to handle spatial data efficiently, as required in computer aided design and geo-data applications (Lawder & King, 2001).

1.1.1. Hilbert Space Filling Curve Technique

The space-filling curve is a 1D curve that uses recursion to cover a certain region(Wu et al., 2020). The Hilbert curve is a one-dimensional representation of a multidimensional space. Such mappings are useful in a variety of applications, including image processing and, more recently, multidimensional data indexing. Given a two-dimensional square space of size N*N, where N = 2n with order n >= 0, the Hilbert curve recursively divides the space into four equal-sized blocks. Each block is given a sequence number which ranges from 0 to N2-1. It's the most well-known and researched space-filling curve. As a method of linearization, we select the Hilbert curve in the space filling curve(Wang et al., 2022). In terms of numerous criteria, it is thought to be extremely local. To create a linear ordering for the datum-point in Key-space by constructing an index to spatial data using a Hilbert Curve of suitable order. This indexing strategy allows the procedure to avoid problems caused by index overlap (Guttman, 1984).

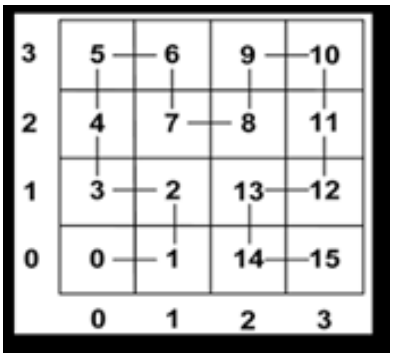
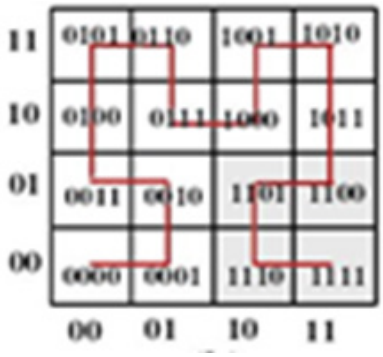


Figure 1: (a) Hilbert Curve Index Value

(Krishnan et al., 2015)



(b) Hilbert Curve Index Binary Value

(Schmidt & Parashar, 2004)

2.Learned Index and Traditional Index

Other types of models, such as deep-learning models—also referred to as learned indexes—can replace index structures. Given that a model can be learnt, taught indexes have several benefits over conventional ones (Manolopoulos, Theodoridis, Tsotras, et al., 2009). This study effectively contrasts standard spatial indexing with spatial learnt indexing. Learned index structures produce an explicit representation of the underlying data in order to provide successful indexing. Numerous spatial indices have been created, and they may be divided into two groups: multidimensional point indices and multidimensional region indices(Nievergelt et al., 1984). Examples of the first category include the LSD-tree(Lomet & Salzberg, 1990), the grid file(Beckmann et al., 1990), the hB-tree (Freeston, 1995), the buddy tree(Sellis et al., 1987), and the BV-tree (Qin et al., 2022). The R-tree (Chen & Chang, 2011) and quad-tree (Li & Feng, 2005), together with their variants, are the second group's most well-known representations. Similar to point indexing, two different approaches for indexing areas (data driven and space driven) have been presented such as R tree (Shah et al., 2021).

3.Spatial Database Query

It is challenging to locate the sought object in a large, muddled dataset, which calls for both computational complexity and time. To address these problems, other study approaches were recommended. Using the K-Nearest Neighbor (kNN) algorithm is the simplest, most effective, and most successful approach of them all. Pattern recognition, text classification, moving object identification, and other applications are possible using this technique (Dhanabal & Chandramathi, 2011). In order to find objects (or combinations of objects) that meet certain spatial relationships with a reference query object, a spatial query is performed to one (or more) spatial relations (or between them)(Mamoulis, 2022).

3.1.Nearest Neighbor Query Types

There are essentially two types of NNQ: those with a structure and those without. The I-Structure based k-NN method represents the training datasets using tree structures. Ball Tree, k-d Tree, Principal Axis Tree (PAT), Orthogonal Structure Tree (OST), Nearest Feature Line (NFL), and Center Line (CL) approaches are examples of structure-based algorithms. II. Simple, k-NN, Condensed NN, Model based k-NN, Ranked NN (RNN), Pseudo/Generalized NN, Clustered k-NN (CkNN), Continuous RkNN, Mutual kNN (MkNN), and Constrained RkNN are examples of non-structure based kNN approaches. The advantages and disadvantages of each form of closest neighbors approach are discussed in Table 1 (Dhanabal & Chandramathi, 2011).

Table 1: Comparison between Different Kind of Nearest Neighbor (Dhanabal & Chandramathi, 2011)

No.	Technic	Concept	Merits	Demerits	App.
1	Ball Tree k nearest neighbor (Moore & Gray, 2003), (Omohundro, 1989)	To improve the speed.	1. Compatible with high dimensional Objects. 2. Represented data are tuned well to structure 3.Simple to implement 4. Especially used for geometric learning.	1. Implementation cost is high. 2. When distance is increased, performance is decreased.	Robotic, vision, speech, graphics.
2	k-d tree nearest neighbor (kdNN) (Sproull, 1991)	To divide the training data sets into two halves.	1. Perfect balanced trees are formed 2. It is fast and simple.	1. Computational complexity 2.Exhaustive search is required 3. Chance of misleading the points as it blindly splits the points into two halves.	Multidimensional data points.
3	Nearest feature Line Neighbor (NFL) (Li et al., 2000)	To have multiple template per class for classification	1. Accurate classification 2. Effective algorithm for small datasets. 3.Ignored information in nearest neighbor are used	1. Chance of failure if the model in NFL is far away from query point 2.Computational Complexity 3. Hard to illustrate the feature point in straight line.	Face Recognition problems
4	Local Nearest Neighbor (Zheng et al., 2004)	To focus on nearest neighbor prototype of query point	1.Overcomes the limitations of NFL	1.Increase in number of computations	Face Recognition
5	Tunable Nearest Neighbor (TNN) (Zhou et al., 2004)	Calculates the distance first and then implements the steps of NFL	1.Effective for small data sets	1.Large number of computations	Bias problems
6	Principal Axis Tree Nearest Neighbor (PAT) (McNames, 2001)	Uses PAT construction and PAT search	1.Good performance 2.Fast Search	1.Computational Time is more	Pattern Recognition
7	k Nearest Neighbor (kNN) (Cover & Hart, 1967)	To find the nearest neighbor based on _k‘ value	1.Training is very fast 2.Simple and easy to learn 3.Robust to noisy training data 4 .Effective if training	1. Biased by value of k 2.Computation Complexity 3.Memory limitation 4.Being a supervised learning lazy	Large sample data

			data is large 5.It is symmetric.	algorithm i.e. runs slowly 5.Easily fooled by irrelevant attributes	
8	Weighted k nearest neighbor (WkNN) (Bailey, 1978)	To assign weights to neighbors based on distance calculated	1. Overcomes limitations of kNN by assigning equal weight to k neighbors implicitly. 2. Uses all training samples not just k. 3. Makes the algorithm global one	1.Computational complexity increases in calculating weights 2.Slow in execution	Large sample data
9	Condensed nearest neighbor (CNN) (Alpaydin, 1997)	To eliminate data sets which show similarity without adding extra information	1. Reduce size of training data 2. Improve query time and memory requirements 3. Reduce the recognition rate	1. CNN is order dependent; it is unlikely to pick up points on boundary. 2. Computational Complexity	Data set where memory requirement is a main concern
10	Reduced Nearest Neighbor (RNN) (Gates, 1972)	To remove patterns which do not affect the training data set results	1. Reduced size of training data and eliminate templates 2. Improved query time and memory requirements 3. Reduced recognition rate	1.Computational Complexity 2.Cost is high 3.Time Consuming	Large data set
11	Pseudo/Generalized Nearest Neighbor (GNN) (Zeng et al., 2009)	To utilize information of (n-1) neighbors also	1.Uses(n-1) classes which consider the whole training data set	1.Does not hold good for small data 2.Computational complexity	Large data set
12	Clustered k nearest neighbor (Yong et al., 2009)	To select the nearest neighbor from the clusters	1.Overcome defects of uneven distributions of training samples 2.Robust in nature	1.Selection of threshold parameter is difficult before running algorithm 2.Biased by value of k for clustering	Text Classification
13	Reverse k nearest neighbor[(Stanoi et al., 2000), (Korn & Muthukrishnan, 2000), (Yang & Lin, 2001), (Dhanabal & Chandramathi, 2011) (Singh et al., 2003),(Tao et al., 2004)]	Objects that have the query object as their nearest Neighbor, have to be found.	1. Approximate results can be obtained very fast. 2. Well suited for 2-Dimensional sets 3. Well suited for finite, stored data sets 4. Provides decision	1. Requires $O(n^2)$ time 2. Do not support arbitrary values of k 3. Cannot deal efficiently with database updates, 4. are applicable only to 2D	Spatial data set
14	Continuous RkNN (Kang et al., 2006)	To monitor the regions upon updates using FUR tree	1. Overcomes the difficulties of using the kNN and RkNN queries on moving objects. 2. Best suited for monochromatic cases	1. Not suited for bichromatic cases 2. Not suited for large population of continuously moving objects. 3. Memory Limitation	Moving object data set
15	Constrained RkNN (Emrich et al., 2009)	To find the RkNN on moving objects based on constrains	1. Communication load is minimized. 2. CRkNN can be applied to both monochromatic and bichromatic cases.	1. Approximate result can be obtained for bichromatic cases.	Moving object data set especially in GPS

3.2. Nearest neighbors Query Algorithm based on Hilbert Curve

Procedure steps of the Nearest Neighbor algorithm in the Hilbert space filling curve data structure is defined in figure 2. Actually the main idea behind the nearest neighbor search is the distances Philosophy, which is in default is Euclidian distance measure. There are different distance measures such as Manhattan, Hamming and Minkowski, etc.

Procedure *All_Nearest_Neighbors_Finding* (DataSet A, DataSet B, Order n_A, Order n_B)
01: **begin**
02: $ANNSet := \phi$; /* $ANNSet$ is the result of $ANN(A, B)$. */
03: Sorting sequence numbers h_A in dataset A; /* Step 1*/
04: /* Data a_i locates at the block with sequence number h_A in the curve of order n_A.*/
05: For each data a_i with sequence number h_A in dataset A do /* Step 2 */
06: **begin**
07: $CNNSet := \phi$;
08: /* $CNNSet$ is the candidate set to store candidate neighbors in dataset B. */
09: if $(n_A - n_B) = 0$ then /* Condition 1 */
10: **begin** $h_C = h_A$; $CNNSet = CNNSet \bigcup \{h_C\}$;
11: For each $DirN \in$ {'S','N', 'E', 'W'} do
12: $CNNSet = CNNSet \bigcup$ *One_Neighbor_Finding* $(h_C, n_B, DirN)$;
13: **end**;
14: if $(n_A - n_B) > 0$ then /* Condition 2 */
15: **begin** $h_C = h_A \times 4^{n_B - n_A}$; $CNNSet = CNNSet \bigcup \{h_C\}$;
16: For each $DirN \in$ {'S','N', 'E', 'W'} do
17: $CNNSet = CNNSet \bigcup$ *One_Neighbor_Finding* $(h_C, n_B, DirN)$;
19: **end**;
20: if $(n_A - n_B) < 0$ then /* Condition 3 */
21: **begin**
22: $TempSet := \{h_A\}$;
23: /* $TempSet$ is the temporary set to store candidate neighbors in dataset A. */
24: For each $DirN \in$ {'S','N', 'E', 'W'} do
25: $TempSet = TempSet \bigcup$ *One_Neighbor_Finding* $(h_A, n_A, DirN)$;
26: For each $h_t \in TempSet$ do
27: $CNNSet = CNNSet \bigcup$
28: $\{h_C \mid h_C$ ranges from $h_C = h_t \times 4^{nB-nA}$ to $h_C = h_t \times 4^{nB-nA} + 4^{nB-nA} - 1\}$;
29: **end**;
30: Sorting and Filtering distinct sequence numbers h_c in $CNNSet$; /* Step 3 */
31: For each h_C in $CNNSet$ do
32: **begin**
33: if the block with sequence number h_c has no data ;
34: else For each data b_j in the block with sequence number h_B in data set B do
35: $NN := \{b_j : \exists b_j \in B, \neg\exists b_k \in B\{dist(q, b_k) < dist(q, b_j)\}\}$
36: **end**;
37: $ANNSet := ANNSet \bigcup \{(a_i, NN)\}$;
38: **end**;
end procedure

Figure 2: Shows nearest neighbor algorithm using Hilbert Curve (Chen & Chang, 2011)

A Hilbert curve recursively divides the space into four equal sized blocks due to the limit of the block capacity. The Hilbert curve never maintains the same direction for more than three consecutive blocks. Take the

quaternary space in figure 3, as an example. The quaternary space is divided into four equal-sized blocks such that four blocks are in the southwest (SW), northwest (NW), northeast (NE) and southeast (SE) directions of the center point P. We call these blocks as the SW, NW, NE, and SE blocks (Chiu et al., 2019).

Algorithm (1) describes the construction of Hilbert Space filling Curve procedure. It shows Hilbert curve constructing steps by steps briefly.

```
Algorithm 1: Constructing Hilbert index
Name: Hilbert index ()
Input: spatial dataset
Output: Generating Hilbert index
Begin:
// Step 1: Using Transformation Table //
            Let L= length of Idx
   For i= 1 to L
          If Idx(i)= 0: then Hb= Hb+0
          If Idx(i)= 1: then Hb= Hb+1
            If Idx(i)= 2: then Hb= Hb+3
          If Idx(i)= 3: then Hb= Hb+2
         Hilb= Hilb+Hb    ,        Next i
// Step 2: Call Rotate-Reflect Procedure //
        Let K= length of Hilb
        For i=1 to K-1
        Let j= i+1
Begin:
  Case (1)
          If K(i)=0 then
          If K(j)=1: then output(j)= 3
          Else if K(j)=3: then output(j) =1     , Next i
          End if
   Case (2)
          Else If K(i)=3 then
          If K(j)=0: then output (j)= 2
          Else if K(j)=2: then output(j) =0      , Next i
          End if
END
```

4.Techniques of System Evaluating

While there are many indicators available for system appraisal, we choose to focus on three key ones here: I-a confusion matrix is a strategy for summarizing findings on a classification task and for assessing the performance of a system's classification algorithm. Roc) curve for receiver operating characteristics. Plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold values will result in the ROC curve. The following formulas for the true positive rate, false positive rate, and specificity: Specificity, false positive rate, and true positive rate. When assessing the effectiveness of a machine learning model based on regression, the R2 score is a crucial indicator. The coefficient of determination is what it is called. It works by figuring out how much variance there is in the predictions that are explained by the dataset. Simply

put, it is the discrepancy between the model's predictions and the dataset's samples. Mean Square Error, part III (MSE), the average squared error, or the discrepancy between the estimated and actual values, is computed.

TPR/Recall/Sensitivity = TP/(TP+FN) ..... Eq. (1),

Specificity= TN/(TN+FP) .....Eq. (2),

FPR=1- Specificity= FP/(TN+FP) ..... Eq. (3) (Son et al., 2021).

5.The Preprocessing Tasks

Before beginning and working on this project, there are some steps that must be taken as preprocessing phases to prepare the data for the purpose of organizing, interpreting, and presenting large amounts of numerical data. It is also important to make the data in a suitable way so that it can be used for analyses and decision-making. Collecting spatial datasets was the initial phase in this paper's preparation of a good dataset (actual geographical dataset) (spatial dataset is in form of latitude and longitude). Latitude and longitude are then converted to (x, y) coordinates. Putting into practice and evaluating as follows:

5.1. Spatial Data Types That Used

We worked on point, line and polygon spatial objects. Each object has different datasets (real data sets), as a first step using point dataset and applying spatial indexing using , Hilbert data structure as traditional indexing. After that implementing nearest neighbor query and then we measured query execution time. In the second step, actually second work, we used machine learning approach in purpose of learning the index of point objects that got by using Hilbert technique, also implementing nearest neighbor query and then calculating query execution time. To comparing and evaluating both methods (traditional and learned). The next objectives of the research are line and polygon objects with their different datasets. We also applied and implemented the work procedure on line and polygon as point object.

5.2. Work Processing Steps

The structure and procedure steps in this work are described briefly in figure 4 and 5 for traditional and learned method respectively. Structure components are: data preprocessing as first step, indexing algorithm for getting index to each spatial objects, NN query and performance comparison metric are the main objectives that mentioned in the diagram. All spatial datasets of point, line and polygon spatial objects were used as system data input separately, converting spatial data which they are latitude and longitude to (x, y) coordinates form. Then for getting indices for all spatial objects the Hilbert Space filling Curve which is spatial indexing technic used. After that implementing nearest neighbor query were done for all spatial data indices.

As a final step execution time of nearest neighbor query found in purpose of evaluating the query performance of all three spatial indices(for point, line and polygon spatial datasets).

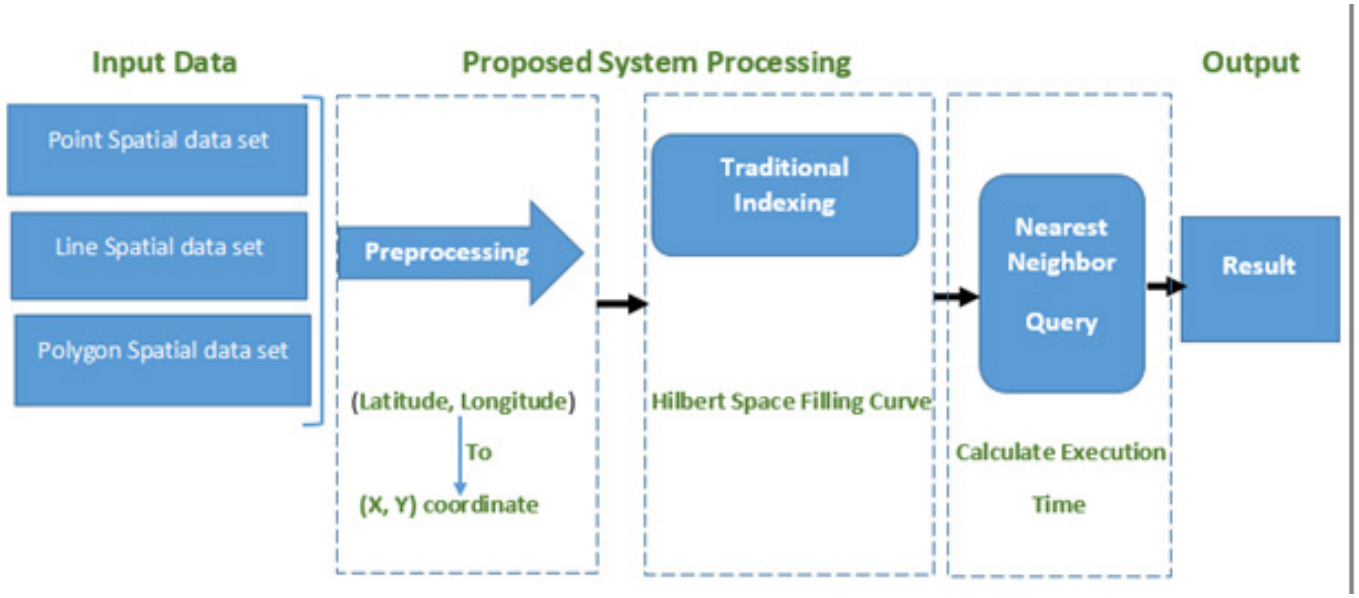


Figure 4: Work Procedure Steps in Traditional Method

As in traditional method, same processes are done. Additionally, each spatial dataset of a point, line, or polygon spatial object was employed as a distinct system data input before being converted from latitude and longitude to (x, y) coordinates. The Hilbert Space Filling Curve, a spatial indexing technique, is then utilized to get indices for all spatial objects. Using machine learning technique, we can then learn our spatial indices. Following that, nearest neighbor queries for all spatial data indices were implemented. Final stage in analyzing the query performance of all three spatial indices was the execution time of the nearest neighbor query (for point, line and polygon spatial datasets). Work steps are shown in image 5.

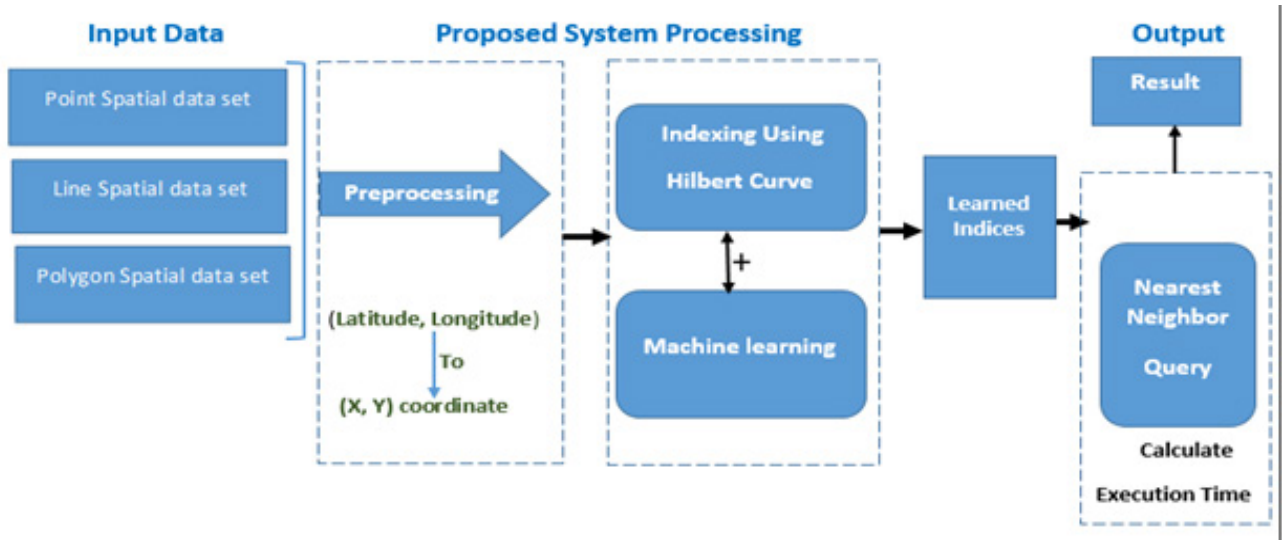


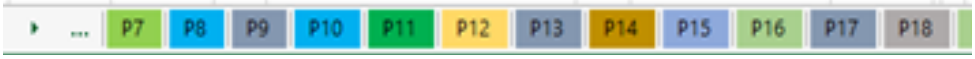
Figure 5: Work Procedure Steps in Learned Method

6-Proposed Algorithm Results

After developing and trying to implement a new, effective spatial indexing technique called the hybrid learning indexing algorithm (HLI), which can be used to index all three types of spatial objects points, lines, and polygons was offered. After determining indices for all spatial objects by using Hilbert Curve algorithm, separately implement nearest neighbor query for each. Means at the beginning we used point index and applied the query at different distances (5, 50, 100, 150, 200, and 250) kilometers respectively for 25 random sample points in purpose of testing. We achieve the following noteworthy and important findings by conducting a nearest neighbor query and calculating execution times in both standard and learnt spatial indices. Table (2) shows test result of finding query execution time just for one point out of 25 point samples of spatial point objects dataset. Results of our study utilizing both conventional and newly taught spatial indices are shown in the tables below. After that we calculated average execution time for all 25 tested points. Table 2 shows tested and implemented results (of point spatial objects just for one sample point).

Table 2: Implementing of NNQ Execution Time of both Methods

Hilbert Curve - Nearest Nighbors		
Query Execution Time at Different Distances in Second(s)	Without ML _ python point sample (14.94,30.5)	With ML _ python point sample (14.94,30.5)
5 km	0.8282	0.31253
50 km	0.8326	0.31682
100 km	0.8375	0.30753
150 km	0.8414	0.30607
200 km	0.8482	0.30542
250 km	0.8826	0.29805



We also applied Nearest Neighbor Query for other spatial objects line and polygon. As in point objects in traditional and learning methods. In traditional method applying NNQ on line objects were done for 25 random sample selected points in various distances (5, 50, 100, 150, 200, and 250) kilometers respectively for comparison issue. Then finding query execution time (ET) for each selected points. Finally average ET were found for all 25. As in traditional, in learnt method, after learning indices also applying NNQ and determining ET for each spatial objects (point, line and polygon) and as well calculating average execution time too. We also used Nearest Neighbor Query for line and polygon spatial objects. Similar to point objects. As an example we just show the one point sample of testing in both methods, the execution time and in various distances. Table three (a) represents all nearest neighbor query execution time of point objects of all 25 selected points

in 6 different distances in traditional method of point spatial objects index and average execution time. And Table three (b) indicates the average execution time of all closest neighbor queries for all 25 chosen points at 6 different distances using the learned (machine learning) approach for point spatial objects.

Table 3: (a) Representation the Tests Table of Traditional Methods

	Traditional Method					
	5km	50km	100km	150km	200km	250km
p1	0.82564	0.82865	0.83465	0.83824	0.84512	0.87694
p2	0.80056	0.80355	0.80756	0.81155	0.81753	0.82017
p3	0.81968	0.82307	0.82754	0.83156	0.83833	**0.84390**
p4	0.81294	0.81605	0.82111	0.82571	0.83247	0.86143
p5	0.81814	0.82109	0.82570	0.83033	0.83828	0.85371
p6	0.82214	0.82512	0.82954	0.83389	0.83951	0.85229
p7	0.82531	0.82831	0.83282	0.83548	0.84040	0.85296
p8	0.83091	0.83439	0.84014	0.84422	0.85153	0.85695
p9	0.82455	0.82754	0.83353	0.83814	0.84500	0.85646
p10	0.83375	0.83704	0.84136	0.84554	0.85341	0.85722
p11	0.86887	0.87275	0.87681	0.87980	0.88512	0.85301
p12	0.88065	0.88564	0.88979	0.89443	0.90076	0.85246
p13	0.81733	0.82033	0.82431	0.82910	0.83509	0.86577
p14	0.80028	0.80330	0.80729	0.81029	0.81427	0.85333
p15	0.81283	0.81582	0.82080	0.82380	0.83078	0.89508
p16	0.79886	0.80389	0.80792	0.81191	0.81690	0.84573
p17	0.80290	0.80588	0.80887	0.81186	0.81486	0.84183
p18	0.79587	0.79987	0.80385	0.80784	0.81484	0.85091
p19	0.79587	0.79790	0.80288	0.80587	0.81186	0.81501
p20	0.78843	0.79142	0.79544	0.79943	0.80601	0.83079
p21	0.79089	0.79388	0.79786	0.80186	0.80584	0.84079
p22	0.82978	0.83274	0.83773	0.84271	0.84870	0.88463
p23	0.82882	0.83181	0.83580	0.84080	0.84577	0.85477
p24	0.83925	0.84256	0.84723	0.85154	0.85881	0.86173
p25	0.82818	0.83257	0.83754	0.84140	0.84818	0.88262
Average Execution Time	0.81970	0.82301	0.82752	0.83149	0.83758	0.85442

(b): Representation the Tests Table of Learnt Methods

	Leaming Method					
	5km	50km	100km	150km	200km	250km
p1	0.306309	0.29922	0.29421	0.30522	0.3052	0.30984
p2	0.29426	0.29932	0.29225	0.30607	0.30555	0.31583
p3	0.32460	0.32660	0.32907	0.32322	0.33038	0.32492
p4	0.31320	0.32257	0.32476	0.31403	0.31774	0.31473
p5	0.31016	0.31470	0.32792	0.32223	0.33580	0.31303
p6	0.30845	0.32615	0.32059	0.32426	0.32123	0.31658
p7	0.32637	0.31786	0.30888	0.31178	0.31495	0.29424
p8	0.31782	0.30719	0.30520	0.32402	0.31087	0.30420
p9	0.30423	0.29625	0.29123	0.30084	0.32550	0.30021
p10	0.32097	0.33348	0.33229	0.30424	0.32454	0.32746
p11	0.30517	0.31080	0.31284	0.29824	0.30423	0.31496
p12	0.30777	0.32250	0.30419	0.30424	0.31084	0.30624
p13	0.31636	0.30794	0.30228	0.31168	0.33242	0.30033
p14	0.30685	0.29822	0.30419	0.33518	0.30486	0.30131
p15	0.30686	0.29660	0.32592	0.33145	0.31082	0.31649
p16	0.31484	0.31184	0.31947	0.32772	0.32207	0.31682
p17	0.31682	0.32562	0.30591	0.30899	0.31285	0.31984
p18	0.30898	0.30224	0.31882	0.30431	0.30328	0.30586
p19	0.30523	0.31087	0.31693	0.31084	0.31297	0.31793
p20	0.29878	0.32819	0.32731	0.33781	0.3135	0.34806
p21	0.32816	0.31484	0.31484	0.32909	0.33156	0.33080
p22	0.31273	0.33074	0.31298	0.32303	0.32787	0.32979
p23	0.33802	0.32846	0.34118	0.34467	0.34106	0.33505
p24	0.33704	0.32659	0.32315	0.31563	0.31204	0.33272
p25	0.31253	0.31682	0.30753	0.30607	0.30542	0.29805
Average Execution Time	0.31370	0.31502	0.31456	0.31699	0.31750	0.31581

Table 4 (a), (b) briefly represents nearest neighbors query execution time (s) of both traditional and learned algorithms using Hilbert Curve technique for point spatial objects respectively which is derived from table 2 and table 3.

Table 4: (a) NNQ Execution Time in Traditional Method for Point Objects

Average Execution Time of Nearest Neighbor Query for Point Spatial Objects **without** Machine learning (Traditional Indexing)	
Distances	Execution Time (second)
5 km	0.81886
50 km	0.82221
100 km	0.82632
150 km	0.82985
200 km	0.83718
250 km	0.85402

In term of comparison between both methods we determining indices for all spatial objects using Hilbert algorithm, and in purpose of evaluating and enhancing spatial performance applying NNQ on all indices of all spatial datasets. Using execution time parameter for nearest neighbor query is the core objective of our work. The most efficient work that done is learning indices and applying NNQ then calculating execution time and finally computing average ET. Testing processes done separately for point, line and polygon. Each has own test tables result .we can just show a brief of the results.

(b) NNQ Execution Time in Learning Method for Point Objects

Average Execution Time of Nearest Neighbor Query for Point Spatial Objects **With** Machine learning (Learnt Indexing)	
Distances	Execution Time (second)
5 km	0.31370
50 km	0.31502
100 km	0.31456
150 km	0.31699
200 km	0.31750
250 km	0.31581

The query execution time varies in traditional and learned indexing algorithms which is clear in figures four. That is considerable difference between both methods and spatial nearest neighbor query processing for point dataset (real dataset), using Hilbert curve technique.

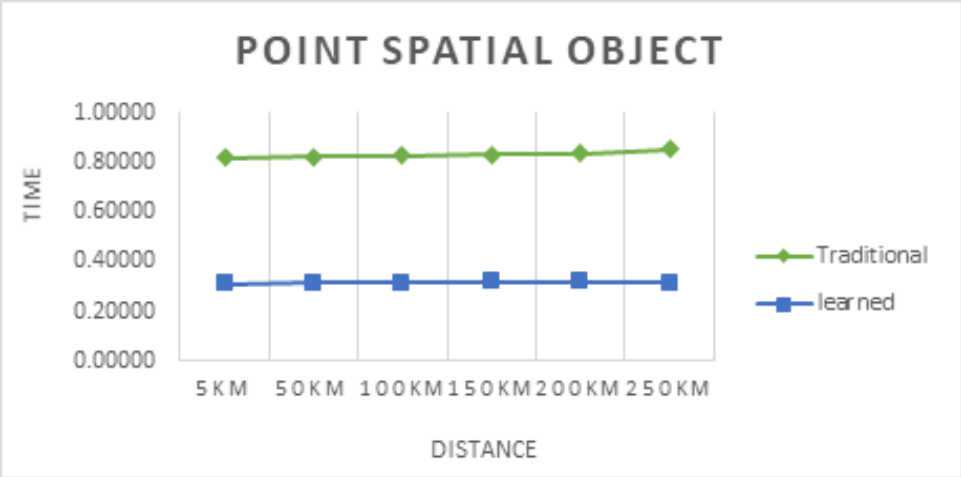


Figure 6: Hilbert, Query Execution Time of learned and Traditional

Figure (٦) shows good response of machine learning technic, that represents less execution time than in Traditional. Less ET in learnt method (with machine learning) gives better performance.

Table five (a), (b) illustrate nearest neighbors query execution time (s) of both traditional and learned algorithms using Hilbert Curve technique for line spatial objects respectively. Tested results of implementing NNQ on line objects in both methods shows best result in leant method (with machine learning), it gives less ET of query. Even in different distances.

Table 5: (a) Traditional algorithm for Line

Line_Average Time_Nearest Neighbors **without** Machine learning	
5 km	0.46509
50 km	0.46546
100 km	0.46694
150 km	0.46846
200 km	0.46993
250 km	0.47159

(b) Learned algorithm for Line

Line_Average Time_Nearest Neighbors **with** Machine learning	
5 km	0.34723
50 km	0.34484
100 km	0.34941
150 km	0.35015
200 km	0.35003
250 km	0.35727

Various query execution time in Hilbert curve technique for line spatial object dataset (real dataset) presented in figure 7(a) and (b) for learned and traditional algorithms.

Figure 7: Query Execution Time of learned and Traditional

As well as in point objects, the testing results in learnt method for line is better than in traditional for line objects. Means the NNQ execution time with machine learning are less than in traditional. Also is the ideal

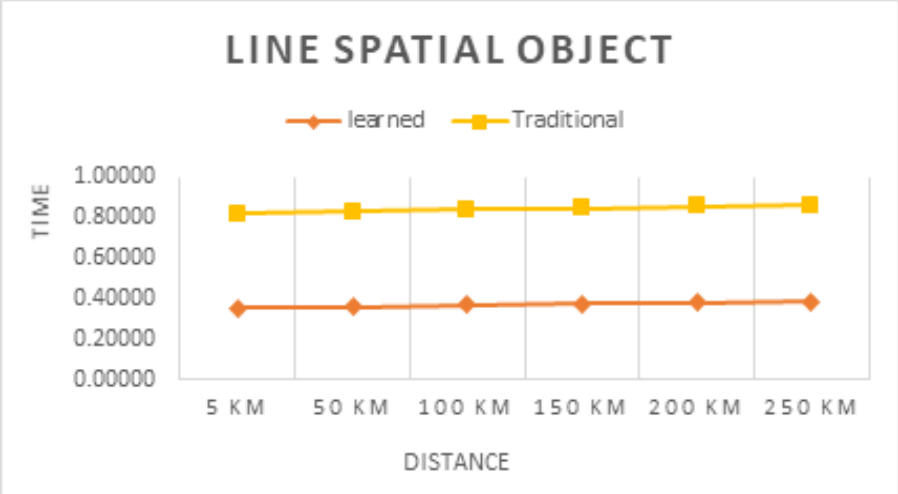


Figure 7: Query Execution Time of learned and Traditional

As well as in point objects, the testing results in learnt method for line is better than in traditional for line objects. Means the NNQ execution time with machine learning are less than in traditional. Also is the ideal result.

Table (6) illustrate nearest neighbors query execution time (s) of both learned and traditional algorithms using Hilbert Curve technique for polygon spatial objects respectively.

Table 6: (a) Traditional Algorithm for Polygon

Polygon_Average Time_Nearest Neighbors without Machine learning	
5km	0.46891
50 km	0.52834
100 km	0.64809
150 km	0.79043
200 km	0.92631
250 km	0.93958

(b) Learned Algorithm for Polygon

polygon_Average Time_Nearest Neighbors with Machine learning	
5km	0.33441
50 km	0.33246
100 km	0.33978
150 km	0.34201
200 km	0.34837
250 km	0.35194

Different query execution time in Hilbert curve techniques for polygon spatial object dataset (real dataset) presented in figure 6 for learned and traditional algorithms.

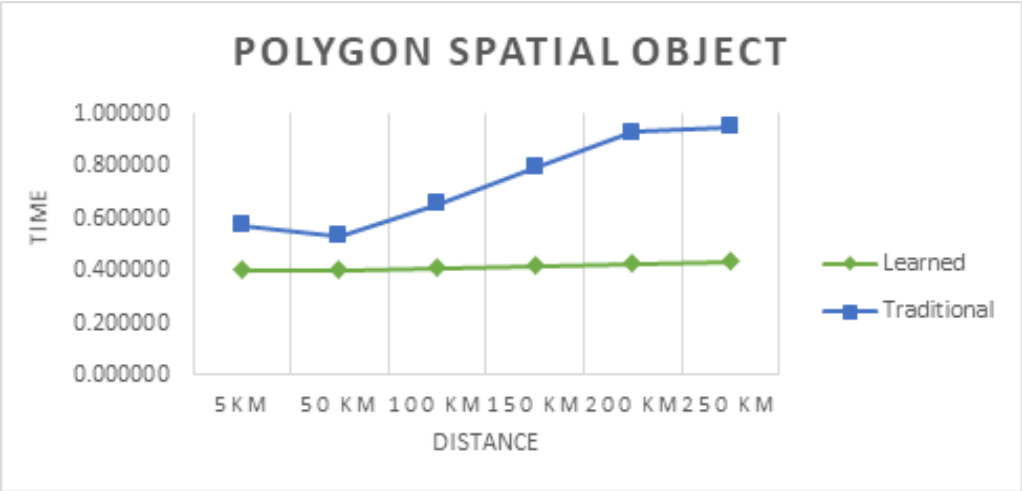


Figure 8: Query Execution Time of learned and Traditional

Figure (8) demonstrates the machine learning technique's effective performance for polygon spatial objects, which takes less time to execute than traditional methods. Better performance is obtained with learned methods that use machine learning and less ET.

7-Models Evaluation Criteria

Remaining issue is evaluating our system by using important parameters. In subsections bellow their role in efficiency and certainty of the proposed system described.

7.1. Using Mean Square Error and Coefficient of Determination

In purpose of evaluating both two algorithms learned and traditional spatial indexing using execution time (s) performance parameter for three spatial data types is the core of our work. In the first step of our work using Hilbert algorithms in case of getting index for each spatial objects types datasets, after that by using both traditional and learned indexing as second step implementing nearest neighbors query for 25 different points in six different distances on all three types of spatial objects. And calculating execution time for both methods (tradition and learned index) in order to evaluate query performance of mentioned methods. Also various performance evaluation indicators were selected for the assessment of the model including coefficient of determination (R2), and mean square error (MSE), their values for point, line and polygon for Hilbert curve are shown below:

Hilbert		
Spatial Objects	Mean Square Error(MSE)	R^2
point	2.36	0.891
line	4.89	0.888
polygon	4.34	0.888

Figure 9: Values of MSE and R2

As represented in figure (9) the values of mean square error (MSE) for all spatial objects are minimum values. And the R2 values are near to 1, which are the optimal and efficient values (the range value of R2 is between 0 and 1).

7.2. Using Receiver Operating Characteristic Curve (ROC- curve)

The best possible prediction method would yield a point in the upper left corner or coordinate (0,1) of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives).

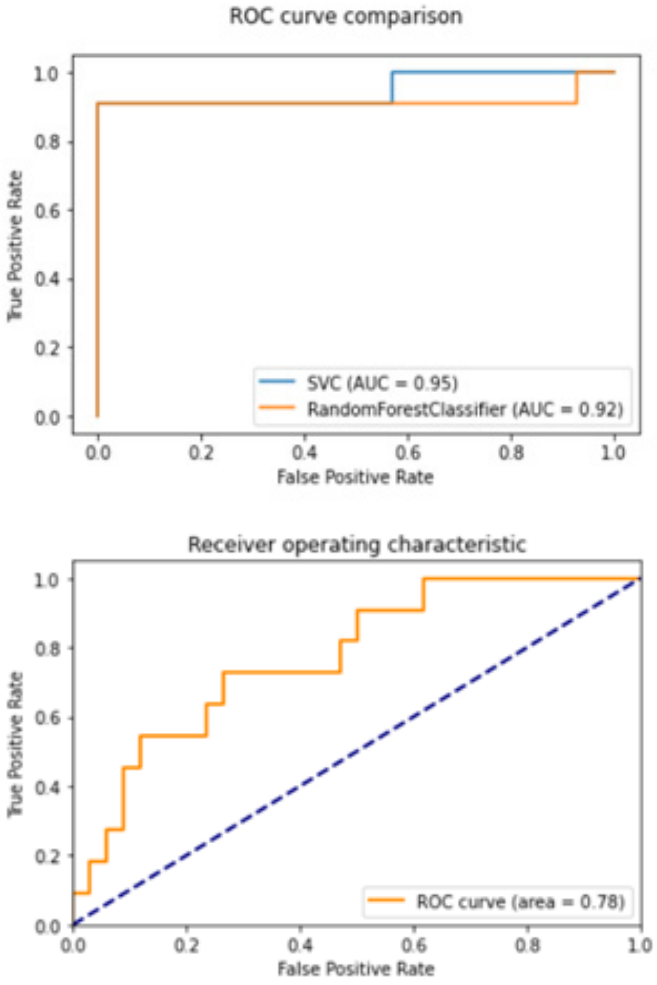


Figure 10: Values of AUC and ROC – curve

8- Conclusion

Our work has proved that spatial indexing based on machine learning, HLI has super advantage over traditional in term of less query execution time for all types of spatial objects. We have reached these conclusions, objects indexing with Hilbert curve has less query execution time in traditional method. But, in learned index in Hilbert curve is faster than in traditional method. For evaluating and enhancing spatial query performance we choose the indexing spatial access method, by indexing objects, can reach any point we request easily and fast. Here we used Hilbert Curve to indexing spatial objects, then applying NNQ for all objects. Also with machine learning approach and learning indices we achieved more efficient responses in all spatial objects in term of less query execution time. This due to enhancing spatial query. Finally the performance of the learned models was challenged using R2, MSE. The value of R2 is efficient in our proposed system, its values are near the 1, which is the significant result of all spatial objects. It is one of the most important evaluation metrics for checking any model's performance. Higher the AUC and also Roc value, the better the model of the system, an excellent model has AUC near to the 1 which means it has a good measure, and we achieved AUC=0.95 in SVM(support vector machine) classifier and AUC=0.92 using randomforest classifier and ROC curve= 0.78, our results are significant and sufficient. Depending on our results from both methods and the performance evaluating measures such as less query execution time in learning method as first point, we achieved significant performance. Then in case of proposed system evaluation we got high R2 values or near to 1 in all spatial objects that are also remarkable. In addition our system evaluators AUC and ROC curve have efficient values. All results can enhance spatial data.

Reference

Alpaydin, E. (1997). Voting over multiple condensed nearest neighbors. Lazy learning, 115-132.

Bailey, T. (1978). A note on distance-weighted k-nearest neighbor rules. Trans. on Systems, Man, Cybernetics, 8, 311-313.

Beckmann, N., Kriegel, H.-P., Schneider, R., & Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. Proceedings of the 1990 ACM SIGMOD international conference on Management of data,

Chen, H.-L., & Chang, Y.-I. (2011). All-nearest-neighbors finding based on the Hilbert curve. Expert Systems with Applications, 38(6), 7462-7475.

Chen, Y., & Patel, J. M. (2006). Efficient evaluation of all-nearest-neighbor queries. 2007 IEEE 23rd International Conference on Data Engineering,

Chiu, C.-Y., Prayoonwong, A., & Liao, Y.-C. (2019). Learning to index for nearest neighbor search. IEEE transactions on pattern analysis and machine intelligence, 42(8), 1942-1956.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), 21-27.

Davitkova, A., Milchevski, E., & Michel, S. (2020). The ML-Index: A Multidimensional, Learned Index for Point, Range, and Nearest-Neighbor Queries. EDBT,

Dhanabal, S., & Chandramathi, S. (2011). A review of various k-nearest neighbor query processing techniques. International Journal of Computer Applications, 31(7), 14-22.

Emrich, T., Kriegel, H.-P., Kröger, P., Renz, M., & Züfle, A. (2009). Constrained reverse nearest neighbor search on mobile objects. Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,

Freeston, M. (1995). A general solution of the n-dimensional B-tree problem. Proceedings of the 1995 ACM SIGMOD international conference on Management of data,

Gates, G. (1972). The reduced nearest neighbor rule (corresp.). IEEE transactions on information theory, 18(3), 431-433.

Güting, R. H. (1994). An introduction to spatial database systems. the VLDB Journal, 3(4), 357-399.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. Proceedings of the 1984 ACM SIGMOD international conference on Management of data,

Jagadish, H. V., Ooi, B. C., Tan, K.-L., Yu, C., & Zhang, R. (2005). iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. ACM Transactions on Database Systems (TODS), 30(2), 364-397.

Jia, L., Liang, B., Li, M., Liu, Y., Chen, Y., & Ding, J. (2022). Efficient 3D Hilbert Curve Encoding and Decoding Algorithms. Chinese Journal of Electronics, 31(2), 277-284.

Kang, J. M., Mokbel, M. F., Shekhar, S., Xia, T., & Zhang, D. (2006). Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. 2007 IEEE 23rd International Conference on Data Engineering,

Korn, F., & Muthukrishnan, S. (2000). Influence sets based on reverse nearest neighbor queries. ACM Sigmod Record, 29(2), 201-212.

Kraska, T., Alizadeh, M., Beutel, A., Chi, E. H., Ding, J., Kristo, A., Leclerc, G., Madden, S., Mao, H., & Nathan, V. (2021). Sagedb: A learned database system.

Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018). The case for learned index structures. Proceedings of the 2018 international conference on management of data,

Krishnan, C. G., Rengarajan, A., & Manikandan, R. (2015). Delay reduction by providing location based services using hybrid cache in peer to peer networks. KSII Transactions on Internet and Information Systems (TIIS), 9(6), 2078-2094.

Lawder, J. K., & King, P. J. H. (2001). Querying multi-dimensional data indexed using the Hilbert space-filling curve. ACM Sigmod Record, 30(1), 19-24.

Li, C., & Feng, Y. (2005). Algorithm for analyzing n-dimensional Hilbert curve. International Conference on Web-Age Information Management,

Li, S. Z., Chan, K. L., & Wang, C. (2000). Performance evaluation of the nearest feature line method in image classification and retrieval. IEEE transactions on pattern analysis and machine intelligence, 22(11), 1335-1339.

Liao, S., López, M. A., & Leutenegger, S. T. (2001). High dimensional similarity search with space filling curves. Proceedings 17th international conference on data engineering,

Liu, X., & Schrack, G. (1996). Encoding and decoding the Hilbert order. Software: Practice and Experience, 26(12), 1335-1346.

Lomet, D. B., & Salzberg, B. (1990). The hB-tree: A multiattribute indexing method with good guaranteed performance. ACM Transactions on Database Systems (TODS), 15(4), 625-658.

Mamoulis, N. (2022). Spatial data management. Springer Nature.

Markl, V. (2000). Mistral: Processing relational queries using a multidimensional access technique. In Ausgezeichnete Informatikdissertationen 1999 (pp. 158-168). Springer.

McNames, J. (2001). A fast nearest-neighbor algorithm based on a principal axis search tree. IEEE transactions on pattern analysis and machine intelligence, 23(9), 964-976.

Moore, A., & Gray, A. (2003). New algorithms for efficient high dimensional non-parametric classification. Advances in Neural Information Processing Systems, 16.

Nievergelt, J., Hinterberger, H., & Sevcik, K. C. (1984). The grid file: An adaptable, symmetric multikey file structure. ACM Transactions on Database Systems (TODS), 9(1), 38-71.

Omohundro, S. M. (1989). Five balltree construction algorithms. International Computer Science Institute Berkeley.

Ooi, B. C., Sacks-Davis, R., & Han, J. (1993). Indexing in spatial databases. Unpublished/Technical Papers.

Qin, D., Wang, H., Liu, Z., Xu, H., Zhou, S., & Bu, J. (2022). Hilbert Distillation for Cross-Dimensionality Networks. arXiv preprint arXiv:2211.04031.

Samet, H. (1990). The design and analysis of spatial data structures (Vol. 85). Addison-wesley Reading, MA.

Schmidt, C., & Parashar, M. (2004). Analyzing the search characteristics of space filling curve-based indexing within the squid P2P data discovery system. Rutgers University, Technique Report.

Sellis, T., Roussopoulos, N., & Faloutsos, C. (1987). The R+-Tree: A Dynamic Index for Multi-Dimensional Objects.

Shah, M. I., Javed, M. F., & Abunama, T. (2021). Proposed formulation of surface water quality and modelling using gene expression, machine learning, and regression techniques. Environmental Science and Pollution Research, 28(11), 13202-13220.

Singh, A., Ferhatosmanoglu, H., & Tosun, A. Ş. (2003). High dimensional reverse nearest neighbor queries. Proceedings of the twelfth international conference on Information and knowledge management,

Son, S.-o., Park, J., Oh, C., & Yeom, C. (2021). An algorithm for detecting collision risk between trucks and pedestrians in the connected environment. Journal of advanced transportation, 2021, 1-9.

Sproull, R. F. (1991). Refinements to nearest-neighbor searching in k-dimensional trees. Algorithmica, 6, 579-589.

Stanoi, I., Agrawal, D., & El Abbadi, A. (2000). Reverse nearest neighbor queries for dynamic databases. ACM SIGMOD workshop on research issues in data mining and knowledge discovery,

Tao, Y., Papadias, D., & Lian, X. (2004). Reverse knn search in arbitrary dimensionality. Proceedings of the Very Large Data Bases Conference (VLDB), Toronto,

Wang, H., Fu, X., Xu, J., & Lu, H. (2019). Learned index for spatial queries. 2019 20th IEEE International Conference on Mobile Data Management (MDM),

Wang, X., Sun, Y., Sun, Q., Lin, W., Wang, J. Z., & Li, W. (2022). HCIndex: a Hilbert-Curve-based clustering index for efficient multi-dimensional queries for cloud storage systems. Cluster Computing, 1-15.

Wu, Y., Cao, X., & An, Z. (2020). A spatiotemporal trajectory data index based on the Hilbert curve code. IOP Conference Series: Earth and Environmental Science,

Yang, C., & Lin, K.-I. (2001). An index structure for efficient reverse nearest neighbor queries. Proceedings 17th International Conference on Data Engineering,

Yong, Z., Youwen, L., & Shixiong, X. (2009). An improved KNN text classification algorithm based on clustering. Journal of computers, 4(3), 230-237.

Zeng, Y., Yang, Y., & Zhao, L. (2009). Pseudo nearest neighbor rule for pattern classification. Expert Systems with Applications, 36(2), 3587-3595.

Zheng, W., Zhao, L., & Zou, C. (2004). Locally nearest neighbor classifiers for pattern classification. Pattern recognition, 37(6), 1307-1309.

Zhou, Y., Zhang, C., & Wang, J. (2004). Tunable nearest neighbor classifier. Pattern Recognition: 26th DAGM Symposium, Tübingen, Germany, August 30-September 1, 2004. Proceedings 26,