4DHI: An index for approximate kNN search of remotely sensed images in Key-Value databases

1st Chamin Nalinda Lokugam Hewage
School of Computer Science
University College Dublin
Dublin, Ireland
chamin.lokugamhewage@ucdconnect.ie

2nd Anh Vu Vo
School of Computer Science
University College Dublin
Dublin, Ireland
anhvu.vo@ucd.ie

3rd Nhien-An Le-Khac
School of Computer Science
University College Dublin
Dublin, Ireland
an.lekhac@ucd.ie

4th Debra Laefer
Center for Urban Science and Progress
New York University
New York, USA
debra.laefer@nyu.edu

5th Michela Bertolotto
School of Computer Science
University College Dublin
Dublin, Ireland
michela.bertolotto@ucd.ie

2022 IEEE International Conference on Cloud Engineering (IC2E) | 978-1-6654-9115-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/IC2E55432.2022.00025

***Abstract*—State-of-the-art, scalable, indexing techniques in location-based image data retrieval are primarily focused on supporting window and range queries. However, support of these indexes is not well explored when there are multiple spatially similar images to retrieve for a given geographic location. Adoption of existing spatial indexes such as the kD-tree pose major scalability impediments. In response, this work proposes a novel scalable, key-value, database oriented, secondary-memory based, spatial index to retrieve the top k most spatially similar images to a given geographic location. The proposed index introduces a 4-dimensional Hilbert index (4DHI). This space filling curve is implemented atop HBase (a key-value database). Experiments performed on both synthetically generated and real world data demonstrate comparable accuracy with MD-HBase (a state of the art, scalable, multidimensional point data management system) and better performance. Specifically, 4DHI yielded 34% - 39% storage improvements compared to the disk consumption of the original index of MD-HBase. The compactness in 4DHI also yielded up to 3.4 and 4.7 fold gains when retrieving 6400 and 12800 neighbours, respectively; compared to the adoption of original index of MD-HBase for respective neighbour searches. An optimization technique termed "Bounding Box Displacement" (BBD) is introduced to improve the accuracy of the top k approximations in relation to the results of in-memory kD-tree. Finally, a method of reducing row key length is also discussed for the proposed 4DHI to further improve the storage efficiency and scalability in managing large numbers of remotely sensed images.**

***Index Terms*—remotely sensed images, approximate k nearest neighbor search, key-value databases, scalability.**

I. Introduction

Advances in spaceborne and airborne image mapping technologies has led to an unprecedented availability of heterogeneous data sets of remotely sensed imagery data (RSID) [1]. For example, in 2013, one of NASA's governing agencies held a total of 7.5 PBs of archived satellite data [2]. Two years late, the average daily archive growth translated to 5.8 PBs annually [1]. In 2019, the China Center for Resources Satellite Data and Application held more than 16 million remote sensing images (RSIs). Such unprecedented availability has generated an increasingly challenging set of storage and querying problems.

To combat this, many researchers are investigating a range of techniques to index and retrieve at scale. Prominent examples include, GeoMesa [3], RASDAMAN [4], TileDB [5], RSMI [6], and the GeoSOT based HBase RSID system [7]. These systems and their corresponding indexing techniques are designed with the objective of coping with sheer volume of data while ensuring commensurate performance for image retrieval at scale. Critically, the indexing techniques for these systems are mainly geared towards supporting window queries and range queries- i.e. retrieving images or image tiles that overlap with a manually selected region. Typically, these images will overlap the initial region of interest at different scales. For example, some of the retrieved images will only cover a subset of the region, while others may partially overlap and fully cover and extend beyond the selected region.

The issue is further complicated by the fact that they can be from different visual perspectives. Furthermore, spatially similar images provide more direct insight on the entire initial region of interest in a straight forward manner. These most spatially similar images will have similar or near similar spatial coverage with respect to the initial region of interest. However, indexing strategies that perform retrieving of most spatially similar images given to an initial region of interest, or in other words, top k nearest neighbours (k-NN) images with respect to a given region have not been studied well in the literature.

A potential reason for the absence for location specific top k-NN search of images can be attributed to not having a scalable index to effectively cope with large amounts of data. A widely employed index for location specific k-NN searching is the kD-tree [8]. By transforming the image bounding box coordinates in the diagonal (i.e. *xMin*, *yMin*, *xMax*, *yMax*) to four dimensional (4D) point values in 4D space, one could

Identify applicable funding agency here. If none, delete this.

978-1-6654-9115-0/22/$31.00 ©2022 IEEE
DOI 10.1109/IC2E55432.2022.00025

develop required spatial indexes with kD-tree. However, since kD-tree is an in-memory spatial index, when dealing with a large number of RSID, kD-tree indexes could lead to excessive main-memory usage.

Importantly, kD-tree construction time is heavily impeded by the large size of RSID, with consequent increase in the response time of the initial top k-NN search query. Additionally, due to the volatile nature of main-memory, the adoption of a kD-tree will also result in regeneration of kD-tree indexes in power failure scenarios. Furthermore, as a kD-tree is sensitive to the order in which the points are inserted, the kD-tree construction time can be further negatively extended by unorderly bounding box insertions. Thus, the adoption of in-memory kD-tree data structure cannot be viewed as a scalable indexing approach for the retrieval of top k-NN images. A potential solution to aforementioned problems would be to employ a secondary-memory index, which can perform k-NN search. Implementation of such a secondary-memory index atop state-of-the-art databases has been shown by Laefer et al. 2018 to radically overcome scalablity issues in aerial laser scans

Key-Value (KV) databases are a class of state-of-the-art NoSQL databases that can be employed as potential databases for the top k-NN search of RSID. Internally, KV databases represent all data as key-value data pairs [9] and store data by mapping their key identifiers to their corresponding data value [10], [11]. KV databases are designed to manage a large amount of data through its distribution across multiple machines. Thus, the adoption of a KV database enables the exploration of horizontally-scaled, shared-nothing-architecture solutions. However, presently Log-Structured-Merge-trees (LSM-trees), which is the underlying data structure in the storage engines of KV databases, do not inherently support nearest neighbour (NN) search. Thus, implementing the secondary-memory implementations of kD-tree such as kD-B-tree [12] and B-kD-tree [13], which are tailored for B-tree based storage engines are not feasible for use in KV databases. To the authors' best knowledge, no secondary-memory implementation of kD-tree is yet available for KV databases that uses LSM-tree underneath its storage engine.

However, a solution that performs NN searches through approximate top k-NN searches of multidimensional point data atop KV databases has been achieved in the in MD-HBase [14]. MD-HBase, is built atop HBase - a leading KV database. It uses two persistent spatial indexes, namely (i) region quadtree, and (ii) a multi-dimensional index that mimics the kD-tree index. The corresponding indexes on MD-HBase are based on the Z-order [15] Space Filling Curve (SFC). The corresponding Z-order index values of multidimensional points are stored as binary values or base 2 values. Since bounding boxes can be indexed using their *xMin*, *yMin*, *xMax*, *yMax* coordinates, the indexing strategy readily available in MD-HBase can be adapted to implement 4D indexes. Thus this seems to hold the potential for a secondary-memory based, scalable solution for location-specific, approximate top k-NN searches of RSID.

However, the emphasis of MD-HBase is on spatial point data management. Furthermore, the alternative of a Hilbert curve [16] provides better packing and organization of rectangles [17]. Therefore, to better understand on which curve to adopt in approximate top k-NN search of RSID, further investigations are needed. This work presented herein investigates an alternative indexing technique for the approximate k-NN search of RSID that are being managed in KV stores by employing a Hilbert SFC. For fair comparison, this work also employs the Hilbert SFC on the four bounding box coordinates of RSID (i.e. *xMin*, *yMin*, *xMax*, *yMax*). Thus, we are presenting a **4D Hilbert index (4DHI)**. Experiments were conducted on both synthetically generated and real world image bounding box data sets on a KV database cluster.

The contributions of this work are summarized as follows:

- A 4DHI - a novel, secondary-memory index structure for approximate k-NN searches of RSID that are managed in KV databases.
- An approximate k-NN search algorithm.
- An evaluation based on a real world data set that shows that the 4DHI and 4D index in MD-HBase provide comparable accuracy in returning similar percentages of approximate k-NN results in comparison to precise results of kD-tree.
- Demonstration of the effectiveness of compact index sizes on storage performance and query response time by representing index keys in base 10 format compared to base 2 format in MD-HBase.
- An optimization strategy termed *bounding box displacement* (BBD) to improve the accuracy of the approximate top k-NN result set.
- A methodical approach to construct more compact index keys to improve the storage of 4D indexes.

The remainder of the paper is organized is as follows. Section II provides an overview on the use of SFCs for non-point spatial data management in KV databases. Section II also shares preliminary results on Z-order curve and Hilbert SFC and provide index technique of MD-HBase in brief. Furthermore, Section II discusses state-of-the-art, scalable indexing techniques adopted in prominent RSID storage and retrieval systems. Section III describes algorithm usage for the construction of the 4DHI index and the approximate k-NN algorithm use for data querying. In Section IVthe experimental results are presented. Finally, Section V concludes the paper.

II. Background and Related Work

A. Non-point spatial data representation in KV databases

Spatial transformation is a technique to obtain the spatial representation of non-point objects [18]. In this technique, the non-point objects are transformed into different representations. Typically, transformation techniques map bounding boxes of non-point objects to points in higher dimensional space [19]. Use of SFCs is a prevalent technique in such transformations.

A SFC is a continuous, surjective mapping from $\mathbb{R}$ to $\mathbb{R}^d$ [20]. SFCs map data in multidimensional space into representational values in a single dimensional space by navigating each data in multi-dimensional space only once in a certain order. There is no perfect mapping to preserve global proximity of the points in the multidimensional space [21]. However, once mapped to single dimension, SFCs are relatively good at preserving proximity at a local level [22]. This means that once the multidimensional data are flattened into single ordinal numbers, they can be incorporated into a list of ordered *key-value* pairs where the localities among data points are preserved [3]. As a result, modern KV database-oriented, multidimensional data management adopts SFCs based values in organizing the data. Two prominent examples are the Z-order curve and the Hilbert curve. These SFCs are widely used in KV database-oriented multidimensional, data management. Figure 1 illustrates the navigation order for the Z-order curve and the Hilbert curve in two-dimensional space.

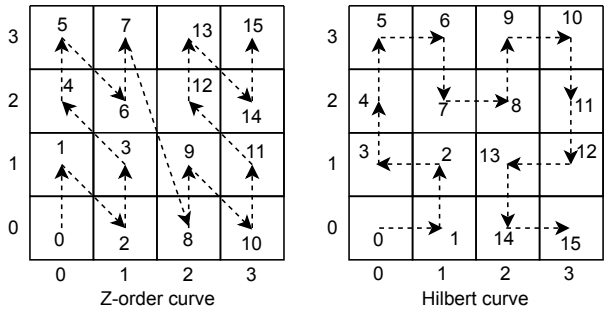

Fig. 1. Navigation order of Z-order and Hilbert curve with their space preservation in two dimensional space.

B. Indexing strategy MD-HBase

MD-HBase employs the trie-based approach for dividing a space into equal-sized grids. Upon division, each dimension's ranges are enumerated using binary values. MD-HBases's indexing layer is based on Z-order values of the dimensions being indexed. Thus, each grid cell in the multidimensional space has an associated Z-order value. These Z-order values in MD-HBase grid cells are computed by interleaving the bits from binary values of different dimensions (Figure 2). This results for each grid cell value. In other words, the Z-order values in MD-HBase are treated as bit arrays in binary format (i.e the keys are in base 2 representation).

C. Scalable RSID systems and their indexing

Efforts to develop novel indexing techniques for remote sensing image retrieval have been undertaken across diverse databases and through varying spatial indexing techniques. In this section, prominent state-of-the-art, database-oriented RSIs storage and retrieval solutions are presented. Their queries of interests, experimental settings, and limitations are highlighted where appropriate.

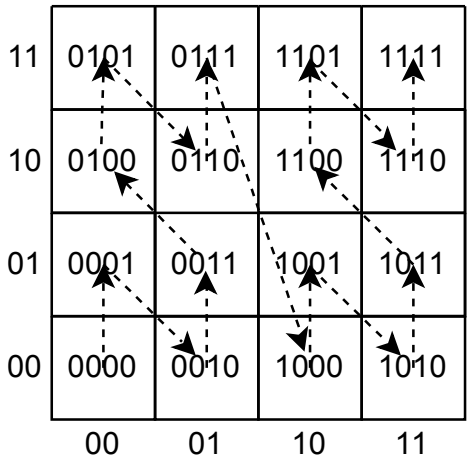

Fig. 2. Binary Z-ordering of MD-Hbase.

RASDAMAN [4], which uses a multidimensional, array data model for processing spatial raster data is one of the leading solutions for storage and retrieval of RSID. When storing images, RASDAMAN divides multidimensional discrete image data into arbitrary array tiles. These subdivided array tiles are later indexed following the built-in, spatial indexes of RASDAMAN such as R+-tree or GiST indexes. Subsequently, they are stored as binary large objects (or BLOBS) inside a PostgreSQL database. While RASDAMAN can be considered a versatile solution for RSID management, the main focus of RASDAMAN has been range queries.

Another novel solution that opens up avenues for RS image storage and retrieval and is based on array data model is TileDB [5]. Similar to RASDAMAN, TileDB represents image data as a 2D array and provides built-in, R-tree index support for data indexing. Moreover, to date, there has yet to be solid experimental evidence to demonstrate the capability of TileDB's native R-tree indexing strategy in retrieving location -specific, spatially similar images at scale.

A more recent work on RS image storage and retrieval is Remote Sensing Images Management System (RSIMS). RSIMS was developed by [6] and is built atop a PostgreSQL database cluster and Ceph distributed object storage file system. At its indexing layer, RSMI employs a distributed multi-level, Hilbert index that employs the latitude and longitude of a single coordinate position. When indexing images, this approach first determines the level at which the spatial index needs to be calculated. The level is based on the spatial coverage or the spatial area of the respective image. Once the level is determined, the corresponding Hilbert code for the image is ascertained through the latitude and longitude of the geometric centre of the image. Similar to previous work discussed, window queries and range queries are the targeted functionality.

Organisation of images into predefined levels and the use of a single point in the image as the index position is

commonly employed in other remote sensing storage and retrieval systems, such as those that adopt key-value databases. Prominent examples include Wang et al. [7] and GeoMesa [3].

In 2015, Wang et al. [7] proposed an HBase-oriented solution to store and retrieve remotely sensed image data. That indexing technique employed was based on the GeoSOT global discrete grid system and a packed Hilbert R-tree. The GeoSOT codes of tiles were calculated based on the latitude and longitude details of the respective grids. While the authors did not specifically state the queries of interest, based on the rationale of GeoSOT - i.e. using hierarchical global subdivision at equal angular increments of latitude and longitude, the prime focus appeared to be supporting window and range queries.

GeoMesa is a leading raster data management solution. While GeoMesa is compatible with a wide range of key-value databases, its debut experiments were focused on raster data management via an Accumulo key-value database. GeoMesa stores images as tiles of different granularities or spatial resolutions. When indexing image tiles, GeoMesa adopts XZ-order curve [23], which is designed for non-zero sized, spatial objects by extending the Z-order curve. However, in GeoMesa, the XZ-curve is still applied based on a specific latitude and longitudinal position in the image tile. In other words, the indices constructed through XZ curve do not explicitly embed the spatial extents of the images into its index representations (i.e. all four coordinates *xMin*, *yMin*, *xMax*, *yMax*). Thus, GeoMesa's indexing strategy, although its based on secondary-memory storage, cannot be applied to perform approximate k-NN search of RSID.

III. Methodology

This section describes the spatial index established to enable the approximate top k-NN queries for RSID in KV stores and the approximate k-NN algorithm employed. For ease of understanding and as a quick reference, mathematical symbols and different notations used in Algorithm 1 and Algorithm 2 are summarized in Table I.

TABLE I
SYMBOLS / NOTATIONS AND THEIR MEANING

Symbol / Notation	Meaning
SET_{RSID}	RSID set to be indexed
i_{img}	ith img record in the SET_{RSID}
b_i	ith bounding box obtained from i_{img} record
$xMin_i$, $yMin_i$, $xMax_i$, $yMax_i$	minimum and maximum coordinates of b_i
$hOrder$	Order of the Hilbert curve
$fourDHI_{10}$	4D Hilbert index in base 10
$SET_{fourDHI_{10}}$	Complete set of 4DHI of the entire SET_{RSID}
Vk	approximate kNN results set
Ck	candidate list of approximate kNN
Qw	querying window
$fourDHI_{qw}$	4D Hilbert value of querying window
$xMin_{qw}$, $yMin_{qw}$, $xMax_{qw}$, $yMax_{qw}$	bounding box coordinates of querying window
$xMin_c$, $yMin_c$, $xMax_c$, $yMax_c$	bounding box coordinates of candidate image
res	candidate row
$fourDHI_{candi}$	4D Hilbert value of candidate row key
ED_{candi}	Eucledian distance of candidate image coordinates with query region coordinates

A. Obtaining Hilbert order for bounding boxes

Hilbert like scanning orders applied on two-dimensional points cannot be applied to other object types such as lines and polygons. As stated previously, the scanning orders for non-point objects on a plane are typically obtained by mapping the bounding boxes of the objects to points in higher dimensions [19]. Following the initial work of Kamel and Faloutso [17], the approach presented herein indexed the bounding boxes of RSID by mapping *xMin*, *yMin*, *xMax*, and *yMax* coordinates to 4D points in a 4D Hilbert space. This approach enables embedding of the spatial extents (i.e. height and width information) of images to its index representation. More precisely, this approach enables the grouping of bounding boxes that are similar in their geographic location and spatial extents into a single-dimensional, sorted order. Furthermore, this facilitates mapping and sorting adjacently on 4D Hilbert curve of location-specific, spatially similar images that reside in close proximity in the real world. Unlike in the case of MD-HBase where the index values are computed as base 2 values, the proposed work computes 4D Hilbert values as base 10 values.

Algorithm 1 (Create 4DHI) shares the steps involved in the calculation of 4DHI for a given RSID set. Algorithm 1 takes three inputs: (i) set of images, (ii) Hilbert order, and (iii) integer value 4 - which is the dimension of the Hilbert curve. At the start of Algorithm 1, the $SET_{fourDHI_{10}}$, which is the set that contains the 4DHI of the entire image data set (i.e. SET_{RSID}), is set to null (Line 2). Subsequently, the looping across the set of input images (i.e. SET_{RSID}) is performed. When looping, for each i_{th} image record, the bounding box coordinates of the i_{img}, i.e. $b_i(xMin_i, yMin_i, xMax_i, yMax_i)$, is ascertained (Line 4). Once the bounding box coordinate for respective image is obtained, the corresponding 4DHI value for i_{img} is calculated via the *Hilbert* function. This *Hilbert* function takes multiple parameters as its input: the bounding box coordinates $b_i(xMin_i, yMin_i, xMax_i, yMax_i)$, Hilbert order (*hOrder*), and integer value 4 - which corresponds to the four dimensional spatial transformation. The function *Hilbert* returns the one-dimensional (1D) real number *fourDHI$_{l0}$* (a base 10 number). This number depicts the 4DHI value of the i_{img} (Line 5). Once 4DHI is established, the value is added to the $SET_{fourDHI_{10}}$. In this work, the 4DHI computation is performed sequentially. Nevertheless, the computation of 4DHI also can be done in parallel.

B. Approximate top k-NN Algorithm

The approximate top k-NN algorithm that is adopted in this work is presented under Algorithm 2. There are three main steps: (i) variable initialization, (ii) candidate row keys selection, and (iii) refinement and retrieval of top k approximate NNs. First, under variable initialization, the sorted

Algorithm 1: Create 4DHI

Input: SET_{RSID}, $hOrder$, 4

Output: $SET_{fourDHI_{10}}$

1 **Function** FourDHI($SET_{RSID}, hOrder, 4$) **:**
2 $SET_{fourDHI_{10}} \leftarrow \emptyset$
3 **foreach** $i_{img} \in SET_{RSID}$ **do**
4 $b_i(xMin_i, yMin_i, xMax_i, yMax_i) \leftarrow i_{img}$
5 $fourDHI_{10} \leftarrow Hilbert(b_i, hOrder, 4)$
6 $SET_{fourDHI_{10}}.add(fourDHI_{10})$
7 **end**
8 **return** $SET_{fourDHI_{10}}$
9 **End Function**

approximate k-NN result set, candidate list of approximate k-NN result set, and 4D Hilbert value of querying window are set to null and zero (Line 2-4). Subsequently, under candidate row keys selection, the 4DHI corresponding to the querying window Qw is obtained from the coordinates of Qw (Line 5). These coordinate values of Qw are already known, as they are provided as inputs to the algorithm (i.e. $xMin_{qw}$, $yMin_{qw}$, $xMax_{qw}$, $yMax_{qw}$). The obtained 4DHI is called the $fourDHI_{qw}$. This $fourDHI_{qw}$ provides the query position or the row key for the KV table scan.

With $fourDHI_{qw}$, the candidate row key set is determined with the input k. The input k provides the upper-bound and the lower-bound for the table scan. The upper-bound includes k records from query position, while the lower-bound includes k records prior to query position (Line 6-7). Thus, the candidate set includes $2k$ records. These $2k$ records are added to candidate set Ck. Importantantly, the proposed Algorithm 2 considers a total of $2k$ rows. However, as will be subsequently demonstrated, the initial observations in Table II in Section IV indicated that an increase in number of rows scanned could improve the accuracy. However, this is not guaranteed and could negatively impact the query response time (see discussion on this point in Section IV under optimization techniques). Thus, when formalizing Algorithm 2 only $2k$ rows were considered.

Step 3, the refinement (i.e Line 8- Line 14), is performed on the obtained candidate set Ck. For each candidate result res, the row key is obtained, which is $fourDHI_{candi}$ (Line 10). Also, for each candidate result res in Ck, the Euclidean Distance (ED) with initial Qw coordinates is acquired. This ED of the candidate record is termed ED_{candi} (Line 11). Subsequently, the algorithm adds $fourDHI_{candi}$ and ED_{candi} pairs to the set Vk that holds the approximated k-NN result set (Line 12). Once all records in Ck are processed, Vk is constructed. Then the pair of records in the Vk are further sorted in ascending order according to ED_{candi} values in each pair (Line 14). Finally, the top k results from set Vk (Line 15) are returned.

Algorithm 2: Approximate_kNN(Qw, k)

Input: Query window Qw ($xMin_{qw}$, $yMin_{qw}$, $xMax_{qw}$, $yMax_{qw}$), integer k

Output: approximate top k-NN: top(k, Vk)

1 **Function** Approximate_kNN(Qw, k) **:**
2 $Vk \leftarrow \emptyset$
3 $Ck \leftarrow \emptyset$
4 $fourDHI_{qw} \leftarrow 0$
5 $fourDHI_{qw} \leftarrow Hilbert$(Qw)
6 $Ck.add(row(fourDHI_{qw}) - num\_rows(k) :$
7 $row(fourDHI_{qw}) + num\_rows(k))$
8 **foreach** $res \in Ck$ **do**
9 $xMin_c, yMin_c, xMax_c, yMax_c \leftarrow res$
10 $fourDHI_{candi} \leftarrow res$
11 $ED_{candi} \leftarrow ED([$ $xMin_c, yMin_c, xMax_c, yMax_c],$ $[xMin_{qw}, yMin_{qw}, xMax_{qw}, yMax_{qw}])$
12 $Vk.add(< fourDHI_{candi}, ED_{candi} >)$
13 **end**
14 $Vk.sort('ED'_{candi})$
15 ***return*** $top(k, Vk)$
16 **End Function**

IV. EXPERIMENTAL EVALUATION

In this section, the implementation of the proposed 4DHI and the MD-HBase's indexes are presented. A comparison of nearest neighbour results are presented with respect to the baseline results of a kD-tree. An extensive performance study between the 4D Hilbert index and MD-HBase's index on synthetically generated and real world data sets are presented. Additionally, an optimization technique on improving the accuracy of the 4DHI result set and another optimization technique on improving the storage performance of the 4DHI are also presented.

A. Study data

The real world data set was comprised of a set of aerial images over a portion of the city of Dublin, Ireland. These were collected during a 2015 airborne LiDAR and imagery mapping excercise [24]. The imagery consists of precisely geo-referenced, fully orthorectified images; oblique images captured from two separate cameras; and RGB and colored infrared (CIR) images taken from a nadir camera. The experiments employed 8,438 real images from this data set. The corresponding ground footprints for each image were calculated from the image metadata files. The ground footprint coordinate values of each image corner point were used to obtain the coordinate values of each image bounding box. These bounding box coordinate values ranged from 314272 to 318002 in the x dimension and 232467 to 236056 in the y dimension in the Irish Grid System. The main aims were (i) to compare the accuracy of the NN result sets of both the

proposed 4DHI index and the MD-HBase's index against the result set of kD-tree and (ii) suggest optimizations.

While this dataset was used to show accuracy and optimizations, to compare scalability between the proposed 4DHI and the MD-HBase's index synthetically generated data sets were employed. Each synthetically generated data set represents minimum bounding boxes of images (i.e. *xMin*, *yMin*, *xMax*, *yMax* coordinates), thus termed as *pseudo bounding boxes*. As the prime purpose of using synthetically generated data sets was to investigate scalability, the synthetic data generation approach did not adhere to a specific coordinate reference system. The coordinates values of each pseudo bounding box were based on integer values ranges. For example, in the x dimension it ranged from 200 to 240 and in y dimension from 900 to 965. For each real image bounding box and pseudo bounding box, the 4DHI and counterpart index of MD-HBase were obtained.

B. Implementation of indexes

When applying different SFCs to a given situation, one of the essential factors was to divide the region of interest into similarly sized raster grids. If the raster grid division is not uniform, the comparison between the results yielding from different SFCs are not comparable. Therefore, the study region was divided into similarly sized raster grids to ensure comparable results obtained from the proposed 4D Hilbert index and the counterpart index of MD-HBase which was based on the Z-order curve .

For the real world bounding boxes, an order 19 grid system (19x19 raster grid) was selected, as this was the largest bounding box coordinate value (i.e. 318002) that could be represented with a 19 bit binary key in base two (e.g. 1001101101000110010). The corresponding 4D Hilbert values were calculated for each bounding box by considering the coordinate values of the bounding boxes. Algorithm 1 presented in Section III was applied to construct 4DHI for image data. As stated previously, the computed 4DHI were in base 10 format. For the calculation of 4DHI an external Java library was employed. The counterpart indexes of MD-HBases were calculated by interleaving bits in the four coordinates. To adhere to the MD-HBases's original implementation, the Z-order index values of MD-HBase were calculated as base 2 indexes.

As the maximum coordinate value in pseudo bounding boxes was 965, when creating the raster grid for synthetic data, a 10 x 10 raster grid was adopted. For the generation of 4D Hilbert indexes and MD-HBases' indexes, a technique matching that applied to the real world data was implemented. Once the index generation was done, the 4DHI keys and MD-HBase's index keys were sorted in ascending order. Next, they were ingested into separate HBase tables by making 4D Hilbert indexes and MD-HBase indexes, as required keys for each data ingestion.

C. Experimental setup

Experiments were performed on the Peel cluster at New York University (NYU). Peel is a high-end 18 node cluster that deploys HBase on top of the Hadoop environment. The hardware and software specifications of each compute node in Peel cluster are as follows: 2x24 - core Intel ® Xeon ® Platinum 8268 CPU @ 2.90 GHz, RAM - 384 GB, disk storage - 8 HDD disks 8 TB each, Hadoop 3.0.0, and HBase 2.1.0-cdh6.3.4. The configured replication factor of HBase was 3 (i.e. each data record gets replicated 3 times).

D. Comparing accuracy of 4DHI

In the tests, first the reliability of using a 4DHI for the representation of image bounding boxes as 4D points for approximate k-NN search was tested. This evaluation was performed on the real world data by comparing the precise NN results produced by an in-memory kD-tree and the approximate k-NN results when employing MD-HBases's indexes. In the comparisons, the error metrics used in kD-tree and MD-HBase's indexes and the 4DHI were identical. For example, as the error metric, the ED error metric was adopted. The ED of respective 4D points in image bounding boxes and initial querying window were calculated.

The comparison followed two methods. In first method, 100 bounding boxes were selected from the real image data set. Each bounding box was subsequently parsed as a 4D vector to a kD-tree data structure from which the top 40 precise NNs were obtained. Subsequently, for each bounding box, r records were scanned in the respective 4DHI and MD-HBase index tables implemented in HBase. Here the r number of records varied depending on the intended $topK$ results. In this case the r number represents 10 rows, 20 rows, and 40 rows for the top 10 results, 20 rows, 40 rows, and 80 rows for the top 20 results, and 40 rows, 80 rows, and 160 rows for the top 40 results. The row keys of r scanned records were inspected. Subsequently, the percentage of overlapping for the top 10, top 20, and top 40 nearest neighbor results (i.e. row keys) with respect to kD-tree results were calculated. This process was repeated six times. Therefore, in total, 600 bounding boxes were randomly chosen and their real NNs were obtained through kD-tree and their approximate NNs determined through 4DHI and MD-HBase indexes. Finally we calculated the average for each overlapping for different percentages ranging from 70%, 80%, 90%, and 100%.

For the second method, a large of set consisting of 600 random bounding boxes was selected. Similar to the first approach, the top precise top 40 NNs were chosen using the kD-tree and the different percentages of overlaps for the top 10, top 20, and top 40, approximate NNs for both 4DHI and MD-HBase's index were obtained. The results are presented in the Table II.

Table II demonstrates a trend for improved accuracy with an increase in the number of rows. This intuitive conclusion is distilled in Table II in a more analytical form:

- When retrieving the top 10 approximate NNs

TABLE II
ACCURACY COMPARISONS BETWEEN MD-HBASE'S INDEX RESULTS AND 4DHI RESULTS WITH RESPECT TO RESULTS OF KD-TREE

Method	k value	Rows (n)	MD-HBase				4DHI			
			Overlap percentage (%)				Overlap percentage (%)			
			70	80	90	100	70	80	90	100
1	10	10	9.8	2.3	0.1	0.0	12.3	4.2	0.5	0.0
		20	31.8	18.5	8.1	2.1	37.7	24.0	11.3	2.3
		40	48.2	36.3	24.2	12.2	58.7	44.7	31.3	16.2
	20	20	7.8	0.8	0.2	0.0	9.7	2.0	0.0	0.0
		40	31.5	17.7	8.2	2.0	37.5	24.0	11.3	2.3
		80	44.3	32.8	21.0	9.3	58.7	44.7	31.3	16.2
	40	40	8.5	1.0	0.0	0.0	8.2	1.0	0.0	0.0
		80	28.7	16.5	6.8	0.8	34.8	15.5	6.0	0.3
		160	42.5	31.0	17.8	5.5	57.2	41.3	23.0	4.8
2	10	10	11.7	3.0	1.5	0.2	14.8	4.7	1.2	0.0
		20	38.7	26.1	12.0	4.2	40.5	25.8	12.7	4.0
		40	55.6	45.0	30.0	16.7	60.2	47.0	31.8	16.7
	20	20	10.2	2.3	0.3	0.0	9.5	2.2	0.0	0.0
		40	38.5	22.5	11.8	2.8	37.2	23.3	9.7	1.8
		80	55.7	40.7	27.8	11.8	57.7	44.0	27.0	10.0
	40	40	9.5	1.7	0.0	0.0	8.2	0.7	0.0	0.0
		80	35.0	21.0	7.2	0.7	35.7	18.0	6.2	0.3
		160	53.8	39.7	25.0	9.7	56.3	40.2	21.7	4.0

  - In method 1, the always yielded better accuracy over MD-HBase's 4D Z-order index. The maximum difference between the accuracies of the two indexes was 10%.
  - In method 2, except for one event, the 4DHI yielded better accuracy. The accuracy difference in the only event where MD-HBases's index outperformed 4D Hilbert index was still less than 1%. In contrast, the maximum difference when 4DHI outperformed MD-HBases's index was nearly 4%.
- When retrieving the top 20 approximate NNs
  - In method 1, except for one instance, the 4DHI yielded better accuracy with a maximum difference of 4%. The accuracy difference between the 2 indexes when the MD-HBase's index outperformed the 4DHI was less than 1%.
  - In method 2, MD-HBases's index outperformed the4DHI in 5 instances with a maximum difference of 2%. In events where 4DHI outperform MD-HBase's index, the maximum difference was 4%.
- When retrieving the top 40 approximate NNs
  - In method 1, except for 5 circumstances, 4DHI yielded better accuracy than counterpart index with a maximum difference of 14%. When the MD-HBase index performed better, the difference was less than 1%.
  - In method 2, MD-HBase's index outperformed the 4DHI in 7 circumstances with a maximum difference of nearly 5%. When 4DHI outperformed the MD-HBase indexing, the maximum difference was only 2%. In these experiments, this was the only event where the proposed index demonstrated a maximum overlapping less than the counterpart index.

These analyses show that in the majority of circumstances, the proposed 4DHI performed better. In particular with respect to obtaining the top 10 and top 20 approximate NNs, the performance gain ranged from 10%-14%. This could be attributed to the fact that Hilbert SFC is better at preserving locality compared to the Z-order curve. However, as the experiments were based on only one real world data set further generalization is not yet possible. Thus this work concludes that both 4DHI and 4D MD-HBase indexes are capable of performing location-specific approximate k-NN for aerial images when managed in KV databases. From a SFC adoption point of view, Hilbert SFC often provided better accuracy.

When the results proposed in the 4DHI did not tally with kD-tree results, a further analysis was undertaken. To do so, a bounding box was arbitrarily chosen from which the top 10 neighbours were obtained, as suggested by the kD-tree. The approximate NN results suggested by 4DHI were also obtained by scanning 10 records. In the top 10 results of the 4DHI, 7 out of 10 matched. Next, the bounding boxes of mismatched results among the two data structures were plotted to obtained their visual semantics (see Figure 3). Figure 3 demonstrates that some results of 4DHI do not comply with the results of kD-tree (i.e. only approximations). However, from a visual point of view, those results of 4DHI are still comparable - or in other words, geographically very close to the initial regions of interest.

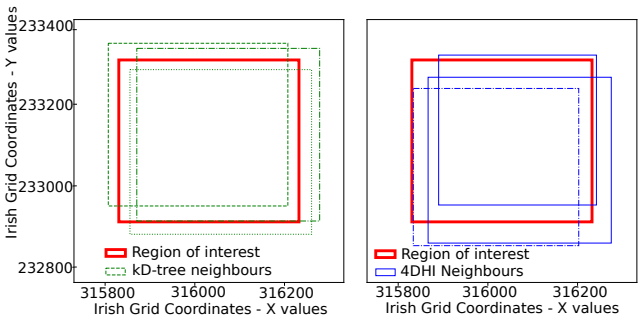

Fig. 3. Analysis of non-overlapping results when using 4DHI with ground-truth results of kD-tree

To deepen the understanding on the approximated results against results of the kD-tree, further analysis was undertaken. Specifically, the ED error values and *Jaccard similarities* between the regions in the two scenarios were measured using the ED error value as the established error metric. The Jaccard similarity metric was used, as it provides a means to measure similarity between polygons pairs. For example, the Jaccard similarity between two polygons can be compared by measuring the ratio between the *Intersection and the Union* of the two polygons as a percentage (i.e. area of overlap / area of union) * 100). The results obtained are presented in Table III.

Table III illustrates that according to the ED error metric, the non-matched, approximated results tend to deviate by a distance value of 20 (i.e. 74 - 54) to 29 (i.e. 96 - 67). Thus, the percentage difference range was 37%, 31%, and 43%. Therefore, even though the non-matching approximated

TABLE III
EUCLIDEAN DISTANCE ERROR VALUES AND JACCARD SIMILARITY IN THE APPROXIMATED RESULTS IN 4DHI IN COMPARISON TO KD-TREE RESULTS

Metric	kD-tree	4DHI	Difference (%)
Euclidean distance of 4D points	[54, 66, 67]	[74, 87, 96]	[37, 31, 43]
Jaccard similarity (%)	[77, 73, 75]	[66, 72, 66]	[11, 1, 9]

results were visually comparable, in the 4D space, these results might deviate from the expected actual results to some extent. However, the matching result set was always greater than the non-matching results. Also, the accuracy of the results set have the potential to be improved through optimization techniques (as will be discussed in Section IV-F). Thus, in overall, this paper claims that the approximated results produce more scalable results for location-specific RSID retrieval.

Moreover, from Table III one sees the Jaccarad similarity differs only to a maximum of 11%. This means that the approximated results still possess reasonably better region overlapping with the union between initial region of interest. This reinforces the claim that the approximated results produce more scalable results for location-specific RSID retrieval.

E. Scalability

Next the scalability of the proposed index was considered - in particular in coping with large amounts of image data, by investigating the storage performance and query response time performance. Scalability was tested using five different synthetically generated data sets, where each data set represents a collection of pseudo bounding boxes. These data sets range from 100K - 1200K (K = 1000), pseudo bounding boxes. For each data set, the corresponding 4DHI were obtained as base 10 values. As the authors of MD-HBase represent their indexes as base 2 binary keys, the counterpart MD-HBase indexes were calculated in base 2. After obtaining the index values, they were injected into 10 different HBase tables by considering row keys as either 4DHI or the counterpart MD-HBase index (5 tables with 4DHI and 5 tables with MD-HBase index). During ingestion the row keys were transformed into their corresponding binary array representations - the format that HBase stores data internally.

1) Storage Performance: For each table, the full disk utilization (including the replication) was obtained. The results in Table IV demonstrate that the proposed 4DHI scales well in terms of storage requirements for high degree of data sizes compared to the storage requirements of MD-HBase's index. When adopting the proposed 4DHI, the users obtained an approximate 34% - 39% improvement in storage capacity vis-a-vis the counterpart MD-HBases's indexes. This improvement can be predominantly attributed to the representation of the index. For example, in the original work of MD-HBase, the authors represented the index keys as base 2 binary values. In this work the index keys were represented as base 10 values, which produces more compact keys - meaning, the lengths of the index keys is much shorter. As a result, less storage is required to store the keys in secondary-memory storage.

TABLE IV
STORAGE CONSUMPTION COMPARISON BETWEEN MD-HBASE'S INDEX AND 4DHI

Num: of pseudo bounding boxes	MD-HBase (MB)	4DHI (MB)	Gain (%)
100K (SET 1)	86.8	54.6	36.6
200K (SET 2)	174.3	110.0	36.8
400K (SET 3)	336.3	207.8	38.2
800K (SET 4)	672.2	442.1	34.2
1200K (SET 5)	1019.9	623.1	38.9

2) Query response time performance: Next, a set of k-NN queries were executed across HBase tables that stored synthetically generated data. This was done to evaluate the versatility of the proposed 4DHI as compared to MD-HBase's indexes. All approximate k-NN queries were based on Algorithm 2. When designing the experiments, four arbitrary bounding box regions (R1- R4) and eight k values, namely: 100, 200, 400, 800, 1600, 3200, 6400, and 128000 were selected as NN values. Each corresponding bounding box region with its respective k-NN value (e.g. R1 with 100 k-NN, R1 with 200 k-NN, etc) was executed 25 times to achieve statistically robust response times. As the focus of these experiments were on the performance of the indexes, when ascertaining query response time, the focus was only on the candidate row selection and the filtering happening on the indexes. Image retrieval I/O time was discarded for both indexes. When analyzing the k-NN response time results, for 100, 200, and 400 k-NNs, the query response times in both cases were always below one second. Thus only comparable values that were above 800 are reported herein. Due to space constraints, only a subset of the overall results are presented, as shown in Table V and Figure 4.

TABLE V
AVERAGE QUERY RESPONSE TIME FOR K-NN QUERIES WHEN ADOPTING MD-HBASE'S INDEX, AND PROPOSED 4DHI FOR APPROXIMATE KNN SEARCH OF BOUNDING BOXES.

Data set	6400 Neighbours			12800 Neighbours		
	Avg: response time (s)		Gain	Avg: response time (s)		Gain
	MD-HBase	4DHI		MD-HBase	4DHI	
SET 1	8.4	2.6	3.2	34.8	10.0	3.5
SET 2	8.2	2.4	3.4	28.6	6.1	4.7
SET 3	8.0	3.7	2.2	28.7	6.1	4.7
SET 4	8.3	5.0	1.7	27.2	11.0	2.5
SET 5	11.0	3.0	3.7	37.9	15.2	2.5

Table V is based on retrieval of k-NN for a pre-selected bounding box region for different sizes of data sets. This demonstrates the impact on k-NN query response time due to the growth in data size with respect to an arbitrary bounding box region. Results shows that the average query response time

for both indexes increased with respect to the growth in data sizes. Nevertheless, compared to MD-HBases's Z-order curve based 4D index approach, the proposed 4DHI yielded greater improvements. Figure 4 is based on data set 5 (i.e. SET 5) - the biggest dataset tested. It demonstrates the average query response times for the four arbitrarily selected bounding box regions for the two indexes.

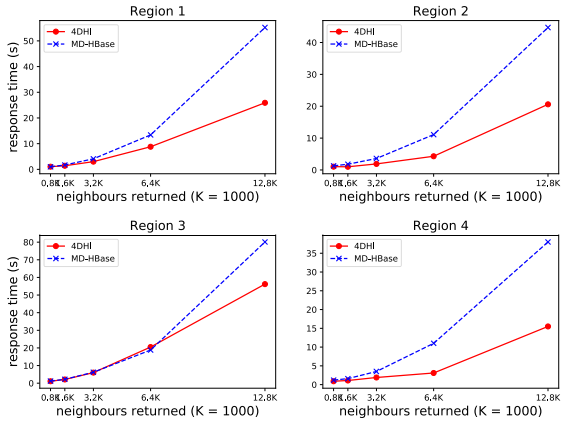

Fig. 4. Approximate k-NN response times for four arbitrary regions selected for SET 5

Similar to the above observation, irrespective of the region, an improvement in query response time is clearly seen when adopting the proposed 4DHI when compared to the MD-HBase's index approach. The improvement shown in Table V and Figure 4, can be predominantly attributed to the compactness of 4DHI. As previously noted, the adopted base 10 format is more compact compared to MD-HBase's base 2 rowkey format. This means, there were less data to be transferred over the network when adopting 4DHI.

F. Optimizations

Finally, two optimization techniques for the proposed 4DHI were explored. These aim at improving accuracy of the approximated results and the efficiency of the storage.

1) Bounding box displacement (BBD):

With the objective of improving the accuracy of the top k-NN result set of the proposed 4DHI similar to the result set of kD-tree, a technique termed - "*Bounding Box Displacement (BBD)*" is proposed. The BBD technique is based on four extra bounding box regions. Each bounding box region in has a similar height and width dimension as that of the initial bounding box. In BBD, the first step is to displace these extra bounding box regions in north-east, south-east, south-west, and north-west directions with respect to the initial region of interest. Subsequently, row scans are performed around the calculated 4DHI of each bounding box, while calculating the ED error values. After this step, the top K result sets from each bounding box region are aggregated to generate a larger candidate set. This expanded candidate set is based on a total of five bounding box regions: initial region and four extra regions as depicted in Figure 5. Finally, similar to Algorithm 2 in Section III, the candidate results are sorted according to their ED values and the top k results are returned.

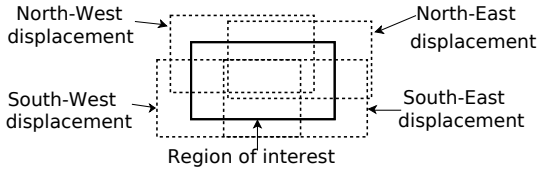

Fig. 5. BBD technique to improve the accuracy of approximate kNN of RSID images

The proposed BBD technique was applied to the real world data. This was done over several regions by employing the approximate k-NN Algorithm 2. In the proposed approximate k-NN algorithm, a total of $2k$ rows are scanned; the term k implies the number of neighbours intends to return. The impact of scanning $4k$ rows was also tested. This was motivated by the initial observation of higher accuracy in approximated results being obtained when more rows are scanned (see Table II).

While employing BBD technique, we displaced additional bounding boxes around the initial region of interest by 5%, 10%, 20%, and 40%. Employing BBD directly increased the accuracy in the final approximated result set for all tested regions. Furthermore, the combination of BBD with an increase in the number of row scans, i.e. $4k$ rows as opposed to scanning of $2k$ rows as stated in the Algorithm 2, also has an impact. For example, with respect to the real world Irish Grid coordinates 315815.56 (i.e. *xMin*), 233211.34 (i.e. *yMin*), 316222.28 (i.e. *xMax*), 233565.4 (i.e. *yMax*) is presented in the Table VI. Figure 6, is based on Table VI; due to space constraints only three neighbour search results are presented from Table VI. The horizontal, dashed line in green in Figure 6 in each sub-graph shows the percentage of overlap with the results set of the kD-tree when not employing BBD when employing Algorithm 2 with a total of $2k$ rows scanned.

TABLE VI
APPLICATION OF BBD TO IMPROVE THE ACCURACY OF 4DHI RESULTS

top k	rows	% of overlap with kD-tree	% of overlap with BBD			
			5%	10%	20%	40%
10	20	90.0%	90 .0%	100.0%	90.0%	90.0%
20	40	80.0%	85.0%	90.0%	80.0%	80.0%
	80	80.0%	90.0%	90.0%	90.0%	90.0%
40	80	72.5%	72.5%	72.5%	77.5%	72.5%
	160	85.0%	87.5%	87.5%	87.5%	90.0%
80	160	76.3%	77.5%	80.0%	77.5%	88.7%
	320	90.0%	90.0%	90.0%	90.0%	100.0%
160	320	84.4%	84.4%	84.4%	84.4%	98.8%
	640	84.4%	98.8%	98.8%	98.8%	98.8%

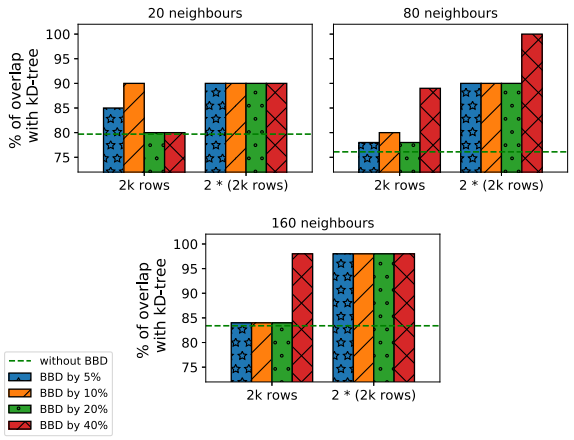

Fig. 6. Impact of BBD on increase in the accuracy in approximated top k results

According to Table VI, initially the corresponding accuracy or the percentage of overlapping with the result kD-tree for top k neighbours ($k = 10, 20, 40, 80, 160$) was always less than 100%. However, with the proposed optimizations improved accuracy up to 100% was achievable, at different displacement percentages. This demonstrated that with the proposed BBD technique, the final approximated result set potentially can be improved, although the final approximated result set will not always improve at all displacement percentages. Even with BBD, the final percentage of overlapping with kD-tree results can be similar to the percentage of overlapping of kD-tree results without the BBD technique. For example, in Table VI for the top 10 approximated neighbours, the percentage of overlap only exceeded 10% of BBD around the initial region of interest. Displacing 5%, 20%, and 40% did not guarantee an improvement. Similar observations are visible for the top 20 approximated results when scanning 40 rows (or $2k$ rows where $k = 20$). More examples can be observed in Table VI.

Identifying the right percentage of displacement is non-trivial, as it can be impacted by several factors such as data distribution. Performing a deep analysis on identifying the right level of displacement factor is beyond the scope of this paper. However, the application of BBD will aim to ensure that the cumulative approximated result be similar or better than the approximated results. For example, in Figure 6, the bar plot heights that show the percentage of overlapping with kD-tree results, always matches or exceed the horizontal dashed line.

The results in Table VI further uncover that the scanning of additional rows might not improve the overall percentage of overlapping with the results of kD-tree for a specific region. This is opposed to the general analysis that our worked observed in Table II. The distinction between the two tables is that, in Table II the statistical results presented accounted for group of regions whereas in Table VI the results accounted only for specific region.

Examples where the increase in the number of rows scan does not always yield better overlapping with results of kD-tree for a given region can be seen in Table VI. In the scenario of top 20 approximated NNs, the increase of scanning of rows from 40 to 80 had no impact on the overall accuracy. However, when couple with use of BBD better accuracy was always achieved irrespective of the percentage of overlap with results of kD-tree. Similar observations can be made with respect to the top 160 approximate NNs in Table VI and the subplot corresponding to the 160 approximate NNs in Figure 6. Thus, adopting BBD in unison with a high number of row scans could improve the accuracy of the approximated results.

Insights on the BBD displacement percentage and increased row scans can also be found in both Table VI and Figure 6. For instance, for smaller k values, such as 20 neighbours, smaller displacements tended to obtain improved accuracy when the number of rows scanned remained at 40 rows (i.e. $2k$ rows). When BBD displaced by 20% or 40%, the accuracy tended to tally with the accuracy obtained without BBD (cf. Figure 6). In contrast, for large number of neighbours such as 80 or 160 with $2k$ rows scanned, the accuracy tended to increase at large displacement percentages of BBD.

BBD introduces additional scanning and filtering steps to the proposed approximate top k-NN algorithm (i.e. Algorithm 2). Intuitively, these additional steps can lead to extra execution time. The experiments conducted while employing BBD technique demonstrated that even for larger k values such as 160 neighbours, the average query response time only increased by 200 milliseconds, even with $4k$ row scans. More specifically, when applying BBD for the approximate top 160 NN search with a total of 640 row scans (i.e. $4k$ rows), the final query response time stayed at 1.6 milliseconds. The query response time for the top 160 approximated NNs without BBD was 1.4 milliseconds. Thus, the impact on query response time from BBD can be considered as trivial in these BBD experiments. Nevertheless, based on the previous experiments on synthetic data, an adverse impact on query response time for k values beyond 1600 can be expected (cf. Figure 4).

2) Compact row key design:

A recent work of the authors of MD-HBase [25] argues that efficient indexing keys need to be short and fixed in their length. In their work, the authors introduce a technique termed reverse computation of prime factorization (RCPF) as a novel SFC technique. This RCPF technique generates more compact index keys termed "P-Indexes". When applied to the real world data here, the keys or the corresponding P-Indexes were indeed significantly small compared to the counterpart 4DHI. For example, when applied to an arbitrary Irish Grid based bounding box where coordinates are 314582 (*xMin*), 232779 (*yMin*), 314898 (*xMax*), 233081 (*yMax*), the P-Index was 268210.2335256357 (17 characters) compared to the counterpart 4DHI of 59118266627987739590792 (23

characters).

Nevertheless when adopting RCPF technique, aggravated index generation times (e.g. 12 seconds to 100 records, 25 seconds to 200 records, 53 seconds for 400 records etc.) were observed. As index generation is a one time operation for a given set of imagery data, arguably the time taken for index generation is a one time cost. Furthermore, although the index is relatively large in size, the index generation time for all real data (i.e. 8438 records) was less than 1 second. Additionally, the top K results' analysis between 4DHI, MD-HBases's index and RCPF index against the in-memory kD-tree result set suggested that both the proposed index and MD-HBase's index performed significantly better at preserving locality of spatially similar images in the 4-dimensional space. For example, both 4DHI and MD-HBases's results were at least 300% better than the RCPF index based k-NN search.

Although the RCPF technique and the P-Index are not suitable for the location-specific scenario, their testing further demonstrated that more compact row keys are favorable, as was shown with the better perfomance of the base 10 oriented indexes compared to the base 2 indexes. As RCPF technique has limitations, in this work a three step technique for generating more compact keys is proposed. The versatility of the proposed optimization technique was demonstrated on a real world data set.

- *Step 1*: remove common prefix from each index key. e.g. in our dataset the lowest 4DHI is 59118266407538384051197 and the highest 4DHI is 59118270076213878661326. Therefore, we propose removing the common prefix - i.e. "*591182*" from all index keys. Once removed, the lowest 4DHI value would be 66407538384051197 and the highest 4DHI value would be 70076213878661326. This step reduces the length of the index to 17 characters - the same number of characters appear when adopting RCPF.
- *Step 2*: recursively divide index keys by prime numbers in their increasing order. At each division, round-up the resulting values to nearest integer. Make sure that each resulting integer is unique and the length of each integer is identical. If not unique, break the recursive operation. e.g. round_up_int (66407538384051197 /2 /3 /5/7) = 316226373257387 and round_up_int(70076213878661326/ 2/3/5/7) = 333696256565054. These new keys have 15 characters.
- *Step 3*: check if Step 1 and Step 2 can be applied recursively on the new representation of keys. e.g. 316226373257387 and 333696256565054 can be further reduced to 16226373257387 and 33696256565054 to more compact keys of length 14 by removing the common prefix "*3*".

The impact of the new compact row keys were tested on both storage performance and k-NN query response time performance. A noted an improvement in storage efficiency was achieved. For example, the 14 character length keys only consumed a total of 32.0 MBs as opposed to the 36.8 MBs consumed by 23 characters length keys. This translated to a storage performance improvement of 13.8% through the proposed three step technique. However, the improvement in the query response time was trivial for the considered k neighbours. Nevertheless, according to the arguments of the authors of the P-Index, compact keys are favorable -in particular with storage efficiency. Thus, we argue that our proposed three step technique on real world data set works better compared to the approach suggested by the authors of P-Index. For example, the proposed technique produced compact row keys with only 14 characters as opposed to 17 characters when applying RCPF technique.

V. Conclusions

This paper presented a 4D Hilbert index for scalable, approximate, nearest neighbour searching of RSID which are managed in KV databases. 4DHI is based on the spatial transformation technique that transforms the minimum and maximum coordinates of image bounding boxes to 4D points in 4D space. Based on 4DHI, a further approximate top k-NN algorithm is proposed that can be employed in KV database environments for location-specific, spatially similar RSID retrieval. When evaluated, the accuracy and the performance of the proposed 4DHI for approximate top k-NN search compared favorably against the results of a MD-HBases's Z-order curve based 4D index. Both 4DHI and MD-HBase's 4D indexes were evaluated based on the ground truth top k-NN results produced by an in-memory kD-tree data structure. The accuracy comparison demonstrated that the proposed 4DHI yielded comparable (if not superior) accuracy compared to the results of MD-HBase's Z-order index.

In terms of storage performance and query response times performance, the proposed 4DHI always out performed counterpart index of MD-HBase, with 34% - 39% storage improvements. Similarly, 1.7 - 3.4 fold gains and 2.5 - 4.7 fold gains were obtained when retrieving 6400 neighbours and 12800 neighbours, respectively. These performance improvements were mainly attributed to the index representation – namely base 10 values as opposed to the base 2 base index keys, as suggested by authors of MD-HBase. Thus, the index keys produced were much more compact compared to the MD-HBase index keys. Similar storage performance and query response times could have been achieved through MD-HBase's indexes, if base 10 representation were adopted.

Finally two optimization techniques for retrieval of location specific spatially similar images were introduced. The first technique, termed as bounding box displacement (BBD). BBD improved accuracy of the final result set at different levels of bounding box displacements around the initial region of interest. The second technique was a three step process that could be adopted to generate even more compact index keys to further improve the storage performance. This proposed three step technique was easy to adopt and time efficient in generating compact indexes compared to the RCPF technique proposed by P-Index work that includes the authors of MD-HBase.

A. Future work

Potential future directions that arise from 4DHI work, in particular in relation to the proposed optimization techniques include the following:

- improve approximate k-NN algorithm further by integrating dynamic BBD displacement percentages.
- integration of number of rows scanned in the query execution as a dynamic value while making sure no impediment on query response time.
- analyze the impact of compact keys arising from second optimization technique on large data sets on query performance.

ACKNOWLEDGMENTS

This publication originated from research supported in part by a grant from Science Foundation Ireland under Grant number SFI - 17US3450. Further funding for this project was provided by the National Science Foundation as part of the project "UrbanARK: Assessment, Risk Management, Knowledge for Coastal Flood Risk Management in Urban Areas" NSF Award 1826134, jointly funded with Science Foundation Ireland (SFI - 17US3450) and Northern Ireland Trust (Grant USI 137). The clusters used for the testing were provided by NYU High Performance Computing Center. The aerial image data of Dublin were acquired with funding from the European Research Council [ERC-2012-StG-307836] and additional funding from Science Foundation Ireland [12/ERC/I2534].

REFERENCES

[1] L. Wang, J. Yan, and Y. Ma, *Cloud computing in remote sensing*. CRC Press, 2019.

[2] M. Chi, A. Plaza, J. A. Benediktsson, Z. Sun, J. Shen, and Y. Zhu, "Big data for remote sensing: Challenges and opportunities," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2207–2219, 2016.

[3] J. N. Hughes, A. Annex, C. N. Eichelberger, A. Fox, A. Hulbert, and M. Ronquest, "Geomesa: a distributed architecture for spatio-temporal fusion," in *Geospatial Informatics, Fusion, and Motion Video Analytics V*, vol. 9473. International Society for Optics and Photonics, 2015, p. 94730F.

[4] P. Baumann, P. Furtado, R. Ritsch, and N. Widmann, "The rasdaman approach to multidimensional database management," in *Proceedings of the 1997 ACM symposium on Applied computing*, 1997, pp. 166–173.

[5] S. Papadopoulos, K. Datta, S. Madden, and T. Mattson, "The tiledb array data storage manager," *Proceedings of the VLDB Endowment*, vol. 10, no. 4, pp. 349–360, 2016.

[6] X. Zhou, X. Wang, Y. Zhou, Q. Lin, J. Zhao, and X. Meng, "Rsims: Large-scale heterogeneous remote sensing images management system," *Remote Sensing*, vol. 13, no. 9, p. 1815, 2021.

[7] L. Wang, C. Cheng, S. Wu, F. Wu, and W. Teng, "Massive remote sensing image data management based on hbase and geosot," in *2015 IEEE international geoscience and remote sensing symposium (IGARSS)*. IEEE, 2015, pp. 4558–4561.

[8] J. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[9] M. T. Özsu and P. Valduriez, *Principles of distributed database systems*. Springer, 2020, vol. 4.

[10] N. Dayan and S. Idreos, "The log-structured merge-bush & the wacky continuum," in *Proceedings of the 2019 International Conference on Management of Data*, 2019, pp. 449–466.

[11] ——, "Dostoevsky: Better space-time trade-offs for lsm-tree based key-value stores via adaptive removal of superfluous merging," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 505–520.

[12] J. T. Robinson, "The kdb-tree: a search structure for large multidimensional dynamic indexes," in *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, 1981, pp. 10–18.

[13] O. Procopiuc, P. K. Agarwal, L. Arge, and J. S. Vitter, "Bkd-tree: A dynamic scalable kd-tree," in *International Symposium on Spatial and Temporal Databases*. Springer, 2003, pp. 46–65.

[14] S. Nishimura, S. Das, D. Agrawal, and A. El Abbadi, "Md-hbase: A scalable multi-dimensional data infrastructure for location aware services," in *2011 IEEE 12th International Conference on Mobile Data Management*, vol. 1. IEEE, 2011, pp. 7–16.

[15] J. A. Orenstein and T. H. Merrett, "A class of data structures for associative searching," in *Proceedings of the 3rd ACM SIGACT-SIGMOD symposium on Principles of database systems*, 1984, pp. 181–190.

[16] D. Hilbert, "Über die stetige abbildung einer linie aufein flächenstück," *Mathematische Annalen*, vol. 38, pp. 459–460, 1891.

[17] I. Kamel and C. Faloutsos, "On packing r-trees," in *Proceedings of the second international conference on Information and knowledge management*, 1993, pp. 490–499.

[18] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Computing Surveys (CSUR)*, vol. 30, no. 2, pp. 170–231, 1998.

[19] H. Haverkort and F. V. Walderveen, "Four-dimensional hilbert curves for r-trees," *Journal of Experimental Algorithmics (JEA)*, vol. 16, pp. 3–1, 2008.

[20] H. Haverkort and F. van Walderveen, "Locality and bounding-box quality of two-dimensional space-filling curves," *Computational Geometry*, vol. 43, no. 2, pp. 131–147, 2010.

[21] R. Laurini and D. Thompson, *Fundamentals of spatial information systems*. Academic press, 1992, vol. 37.

[22] J. Wang and J. Shan, "Space-Filling Curve Based Point Clouds Index," *Proceedings of the 8th International Conference on GeoComputation University of Michigan*, pp. 1–16, 2005. [Online]. Available: http://www.geocomputation.org/2005/WangJ.pdf

[23] C. BÖxhm, G. Klump, and H.-P. Kriegel, "Xz-ordering: A space-filling curve for objects with spatial extension," in *International Symposium on Spatial Databases*. Springer, 1999, pp. 75–90.

[24] D. Laefer, S. Abuwarda, A. Vo, L. Truong-Hong, and H. Gharibi, "2015 Aerial Laser and Photogrammetry Survey of Dublin City Collection Record," 2017, (Last accessed by 20/10/2019). [Online]. Available: https://geo.nyu.edu/catalog/nyu_2451_38684

[25] J. Liu, S. Nishimura, and T. Araki, "P-index: A novel index based on prime factorization for similarity search," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2019, pp. 1–8.