AN EFFECTIVE APPROACH TO PREDICTING LARGE DATASET IN SPATIAL DATA MINING AREA

BY

GRACE SAMSON (u1251405)

A research work submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy in COMPUTER SCIENCE

SUPERVISOR: ***PROF. JOAN LU***

School of Computing and Engineering

Department of Informatics

APRIL 2020

ABSTRACT

Due to enormous quantities of spatial satellite images, telecommunication images, health related tools etc., it is often impractical for users to have detailed and thorough examination of spatial data (S). Large dataset is very common and pervasive in a number of application areas. Discovering or predicting patterns from these datasets is very vital. This research focused on developing new methods, models and techniques for accomplishing advanced spatial data mining **(ASDM)** tasks. The algorithms were designed to challenge state-of-the-art data technologies and they are tested with randomly generated and actual real-world data. Two main approaches were adopted to achieve the objectives (1) identifying the actual data types (***DTs), data structures*** and ***spatial content*** of a given dataset (to make our model versatile and robust) and (2) ***integrating these data types*** into an appropriate **database management system (DBMS) framework, for easy management and manipulation.** These two approaches helped to discover the general and varying types of patterns that exist within any given dataset non-spatial, spatial or even temporal (because spatial data are always influenced by temporal agents) datasets. An **iterative method** was adopted for system development methodology in this study. The method was adopted as a strategy to combat the irregularity that often exists within spatial datasets. In the course of this study, some of the challenges we encountered which also doubled as current challenges facing spatial data mining includes: (a) time complexity in availing useful data for analysis, (b) time complexity in loading data to storage and (c) difficulties in discovering spatial, non-spatial and temporal correlations between different data objects. However, despite the above challenges, there are some opportunities that spatial data can benefit from including: Cloud computing, Spark technology, Parallelisation, and Bulk-loading methods.

Techniques and application areas of spatial data mining (SDM) were identified and their strength and limitations were equally documented. Finally, new methods and algorithms for mining very large data of spatial/non-spatial bias were created. The proposed models/systems are documented in the sections as follows: **(a)** Development of a new technique for parallel indexing of large dataset (*PaX-DBSCAN*), **(b)** Development of new techniques for clustering (*X-DBSCAN*) in a learning process, **(c)** Development of a new technique for detecting human skin in an image, **(d)** Development of a new technique for finding face in an image, **(e)** Development of a novel technique for management of large spatial and non-spatial datasets (aX-tree).

The most prominent among our methods is the new structure used in **(c) above -- packed maintained k-dimensional tree (P*mkd-tree***), for fast spatial indexing and querying. The structure

is a combination system that combines all the proposed algorithms to produce one solid, standard, useful and quality system. The intention of the new final algorithm (system) is to combine the entire initial proposed algorithms to come up with one strong ***generic effective tool for predicting large dataset SDM area, which it is capable of finding patterns that exist among spatial or non-spatial objects in a*** **DBMS**. In addition to **P*mkd-tree*,** we also implemented a novel spatial structure, packed quad-tree **(*Pquad-Tree*),** to balance and speed up the performance of the regular quad-tree. Our systems so far have shown a manifestation of efficiency in terms of performance, storage and speed. The final **Systems (P*mkd-tree* and *Pquad-Tree)*** are generic systems that are flexible, robust, light and stable. They are explicit spatial models for analysing any given problem and for predicting objects as spatially distributed events, using basic SDM algorithms. They can be applied to pattern matching, image processing, computer vision, bioinformatics, information retrieval, machine learning (classification and clustering) and many other computational tasks.

Keywords:

Spatial data-mining, Temporal data-mining, Data types, Machine learning, Pattern recognition, Image processing, Skin detection, Classification, Algorithms, Computer vision, Database management systems (DBMS), Parallel computing, Face detection, DBSCAN, Data structures, Clustering, Spatial data analysis, Data modelling.

QUOTES:

"Everything around us including humans occupy space; therefore, can be likened to an object in space."

"Most big data are spatially referenced''

"Fundamentally, any form of geographical image (ecological, biological, physical, environmental...) with inherent **coordinate points** or **location, can be seen or used as spatial data**''

"We build GIS systems to support **the manipulation of large geographic data,** related to operations such as **digitization**, **visualization** and **analysis"**

ACKNOWLEDGEMENT

DEDICATION:

I dedicate this project work to GOD almighty for making all things possible for us. In addition, I dedicate the work to my family especially, for their immeasurable support throughout the course of the program.

Acknowledgements:

I convey my profound gratitude and special thanks to the **ALMIGHTY GOD** who Himself arranged the process of me taking part in this **Ph.D. programme**. I also love to extend my sincere heart-felt gratitude to the management and staff of **UNIVERSITY OF ABUJA, Nigeria** for their uttermost support both financially and otherwise, I am indeed very grateful and indebted to you. To the management and staff of TERTIARY EDUCATION TRUST FUND **(TETFUND - NIGERIA).** You made this journey financially possible and I am sincerely grateful.

I sincerely recognise and acknowledge the entire management and staff of the school of Computing and Engineering, Dept. of Informatics, for all their kind support and effort in the course of my Ph.D. degree research programme, at the **University of Huddersfield, UK.**

My most sincere gratitude goes to my **supervisor**, **Prof. Joan Lu,** who began this great journey to a glorious research future for me. If I say I am not grateful it would be an understatement, I appreciate all your kind efforts and patience toward the successful completion of this research project.

To everyone whom I came across in Huddersfield, our path has been destined to cross, therefore, I never regret meeting any one of you, else your purpose in my life will not be accomplished, it's you that I need to learn the lessons that I have to, thank you so much for being used to help me through.

Finally, and most importantly, I do acknowledge you Heaven, Zenith, Royal and Regal, my indescribable stars, you guys are really the stars that give light to my path, I love you so much my awesome kids and thanks for your 48 months patience, God bless you. And to my husband, if it were not you, then I would not be here, but because it's you we have seen the end of this unfathomable journey together, thank you for your love and patience and especially for devoting your time to care for the kids, and support my studies. I love you so much.

TABLE OF CONTENT

Contents

List of Figures:

List of Tables

List of Flowcharts

TERMINOLOGIES

Spatial Database Systems	-------	**SDBS**
Spatial Data Mining	-------	**SDM**
Knowledge Discovery in Databases	-------	**KDD**
Euclidean spaces	-------	**ES**
Real coordinate space	-------	**RCS**
Nearest neighbour	-------	**NN**
K-Nearest neighbour	-------	KNN
Minimum bounding rectangles	-------	**MBR**
Spatial database management system	-------	**SDMS**
Spatial Database	-------	**SD**
Data Mining	-------	**DM**
Database Management Systems	-------	**DBMS**
Sort-tile-recursive	--------	**STR**
Minimum bounding rectangles	--------	**MBR**
Geographic Information Systems	-------	**GIS**
Database Systems	--------	**DS**
Density Based Spatial Clustering of Application with Noise		**DBSCAN**
Database Management	-------	**DBM**
Big Spatial Data	-------	**BSD**
Spatial data types	-------	SDT
Packed maintained k-dimensional tree	-------	**Pmkd-tree**
Packed quad-tree	-------	**Pquad-tree**
Data management	-------	**DMGT**
Multidimensional Scaling	-------	**MDS**
Spatial Data	-------	**S**
Data types	-------	DT
Spatial Data Analysis	-------	**SDA**

PUBLICATIONS

PUBLICATIONS:

The papers listed below are the results of the work conducted in the course of this Doctor of Philosophy Degree:

JOURNAL ARTICLES:

Samson, G. L., Lu, J. and Xu, Q. (2017) Large Spatial Datasets: Present Challenges, Future Opportunities. Int'l Conference on Change, Innovation, Informatics and Disruptive Technology, ICCIIDT London – UK, 2016. Available at: http://proceedings.sriweb.org/repository/index.php/ICCIIDT/icciidtt_london/paper/view/24

Samson, G. L., Mistura *M. Usman*, *Aminat A. Showole, Joan Lu*, *Hadeel Jazzaa* (2018) "Large Spatial Database Indexing with aX-tree", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 3, Issue 3, pp.759-773, March-April.2018. http://ijsrcseit.com/CSEIT1833236

***Samson, G. L.**, &_Lu, J. (2018) "*Spatial Clustering in Large Databases Using Packed X-tree" Egyptian Computer Science Journal. ISSN: -1110-2586, Vol. 42, No.2. pp.68-79. http://ecsjournal.org/Archive/Volume42/Issue2/6.pdf

Books Chapters:

Samson, G. L., Lu, J., Usman, M. M., & Xu, Q. (2017). Spatial Databases: An Overview. In J. Lu, & Q. Xu (Eds.), Ontologies and Big Data Considerations for Effective Intelligence (pp. 111-149). Hershey, PA: IGI Global. doi:10.4018/978-1-5225-2058-0.ch003. Available at: http://www.igi-global.com/chapter/spatial-databases/177391

Samson, G. L., & Lu, J. (2016). PaX-DBSCAN: A PROPOSED ALGORITHM FOR IMPROVED CLUSTERING. In M. R. Pańkowska (Ed.) Studia Ekonomiczne. Zeszyty Naukowe, (269524th ed.). Katowice: Wydawnictwo Uniwersytetu Ekonomicznego w Katowicach. Retrieved from www.sbc.org.pl/Content/269524

UNDER REVIEW:

Samson, G. L., & Lu, J. (2016). Novel multidimensional spatial model for fast and accurate human skin detection. Submitted to International Journal of Computer Vision, Springer, May 2019.

Samson, G. L., & Lu, J. (2016). Fast novel clustering algorithm for human face detection. Submitted to The 23rd int'l Conference on Image Processing, Computer vison & Pattern recognition [IPCV'19], in The 2019 World Congress in Computer Science, Computer Engineering, & Applied Computing.

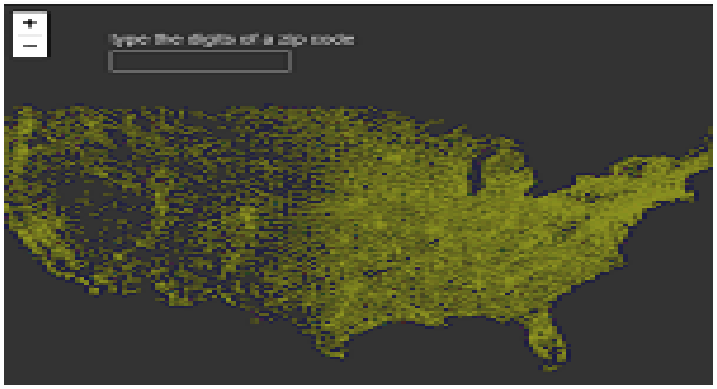
CHAPTER 1. INTRODUCTION:

Due to enormous quantities of spatial satellite images, telecommunication images, health related tools etc., it is often impractical for users to have a detailed and thorough examination of spatial data (S). The study of big data analytics is concerned with the processing of data that consist greatly of diverse spatial locations as we can see from Figure 1. Thus, to predict events that occur at each of these specific geometric or geographic locations is crucial in numerous application domains. Typical problems, requiring the task of location prediction are

- ***Cellular networking***
- ***Crime analysis,***
- ***Demonstrating natural disasters like vegetation diseases, droughts,***
- ***Action (events) or object prediction***
- ***Earthquakes***

Shekhar et al., (2005) emphasized on the huge advances in S and pervasive use of Spatial Databases (SDs). They highlighted the necessity for the automatic acquisition of spatial knowledge. The definition above inspires our research interest. In very recent times, large and small organizations have an inheritance of operational databases (spatial or non-spatial databases) that are always connected to a spatial meaning. These organizations access several databases such as **statistical information for planning, education**, **security, image, multimedia, census, industry, economic, medicine, decision, intelligence and policy making**. The tremendously large size of these datasets makes it challenging to seek for significant designs or connections between these data; therefore, this poses great challenge and difficulty in making useful formal/organizational policies and decisions. Spatial data mining (**SDM)** is the course of ascertaining remarkable and possibly useful but formerly unknown and non-trivial patterns from large organizational or geographic spatial datasets. Particular features of S that impede the application of general-purpose data mining (DM)| algorithms include extended spatial objects, **implicit spatial associations between the elements**, non-**independent observations** and **spatial autocorrelation** between the features. Figure 1 shows some examples of a spatial dataset.

(a)



(b)

Figure 1: (a) Various postcode locations (centres in the UK), represented with point spatial object (b) point spatial objects consisting of 40, 300 points representing various postcode locations in U.S.A

The occurrence of spatial auto-correlation according to **Samson et al., (2014),** coupled with the notion that **continuous data types frequently occur in S** (see Figure 1) makes it imperative to develop techniques, tools, methods, and algorithms for mining spatial patterns from a multifaceted spatial dataset. In addition, S contents according to **Li et al., (2006), Densham and Goodchild (1989),** defined the specific geometric/geographic orientation of the spatial object as it relates to real world S distribution. *S comprises attributes of location, space elements, the figure and their common relationships, and envelops the whole "large-medium-small" level. S may* be the value of polygon (Figure 26 (b); Figure 10)), elevation, road extent, point, region, and building volume. Similarly, *S* can equally be the sequence of geometric names and explanation. It might be spatial relationships, other multimedia and images, graphics, and/or other arrangements of geometric elements. Therefore, the prevalent use of SDs and the rapid development of S, peaks the necessity for the automated discovery of spatial knowledge. SDs are enhanced for handling data, which are saved depending on some geometric/geographic attributes and spatial inference, by considering the data object as an underlying ***n*** dimensional feature space. That means, practically any object

can be saved in a spatial database as long as their individual components can be interpreted in terms of spatial characteristics. Through high level scalability, researchers have proposed several methods and techniques for building and managing spatial structures. Towards this effect, this project has focused on ways of improving these existing systems and proposing novel techniques for mining knowledge and information from spatial databases effectively. Though large spatial datasets are easily characterised by ***query processing***, ***language***, ***indexing***, and ***visualization (***four major paradigms***),*** we have only focused mainly on the issue of indexing and querying with a little touch on visualization. Many researchers have succeeded in proposing several spatial indexing structures, however, the existing structures and most of their variants are limited by the occurrence of storage complexity, speed ambiguity and poor user experience.

This study examines the potentiality of SDM as a tool for the study and analysis of the distribution, spatial organization and location of any form of activities across and around the earth, across diverse platforms and across various disciplines. including **(a)** development of a new technique for detecting skin in an image, **(b)** development of a new technique for finding face in an image, **(c)** development of a new technique for indexing large spatial and non-spatial datasets (aX-tree), **(d)** development of a new techniques for clustering (*X-DBSCAN*) in a learning process and **(e)** development of a new technique for parallel indexing of large dataset (*PaX-DBSCAN*), using spatial modelling techniques.

These new methods and algorithms show huge efficiency and improvements as equalled most futuristic techniques. They achieved an improvement on the effectiveness, feasibility, scalability

They achieved an improvement on the effectiveness, feasibility, scalability and usefulness of existing systems. Our methods benefit from robust partitioning strategies, which minimises the excess and deep partitioning strategy common in most existing systems.

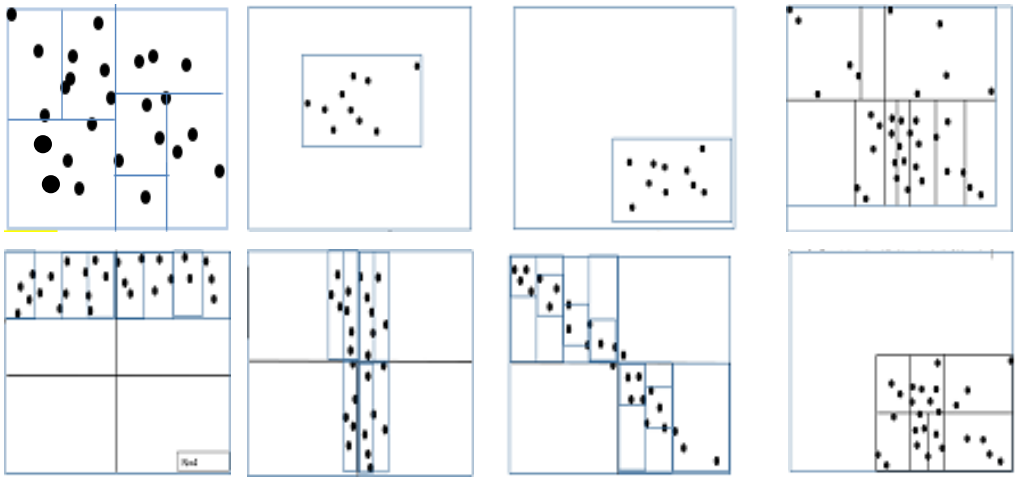


Figure 2: Different partitioning based on geometric location of spatial objects (points)

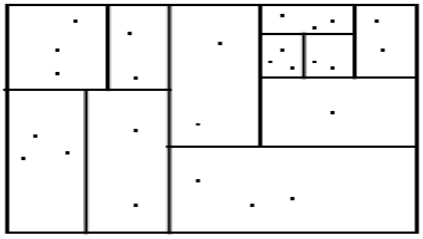


Figure 3: Example space partition where each bucket (MBR) is designed to hold not more than three points.

The partitioning algorithms Figure 2 and Figure 3: rely on a pre-determined approach for partitioning a set of events set or search space into smaller events of subsets, so that each problem (or subset) is solved in the memory. We carried out experimentation and theoretical analysis of large datasets (both real-world and simulated) and the result shows a massive improvement as regards older systems. The most prominent among our methods is the new **packed maintained k-dimensional tree (P*mkd-tree***) for fast spatial indexing and querying. The structure is a combination system that combines all the proposed algorithms to produce one solid, standard, useful and quality system. The intention of the new final algorithm (system) is to combine the entire initial proposed algorithms to come up with one strong ***generic effective tool for predicting large dataset in the SDM area, which is expected to find patterns that exist among spatial or non-spatial objects in a DBMS***. In addition to **P*mkd-tree*,** we also implemented a novel spatial structure, packed quad-tree **(*Pquad-Tree*),** to balance and speed up the performance of the regular quad-tree. Both tree structures are explained in section 5.5. Auspiciously**,** our system has so far shown a manifestation of efficiency in terms of performance, storage and speed. The final **Systems (P*mkd-tree* and *Pquad-Tree*)** are generic systems that are flexible, robust, light and stable. They are explicit spatial models for analysing any given problem and for predicting objects as spatially distributed events, using basic SDM algorithms. They can be applied to pattern matching, image processing, computer vision, bioinformatics, information retrieval, machine learning (classification and clustering) and many other computational tasks.

1.1 Rationale:

Objects like medical or satellite image data, very large-scale integration (VLSI), biometric and human related data, geographic (map) data, computer-aided design data or any other form of object, in one way or the other contain spatial-related information. To extract their spatial integrant, these objects may be represented on the computer in either 1) **Raster format**, made up of *n-dimensional* pixel maps or bitmaps. For instance, a 2D image (pictures, satellite) can be portrayed

as raster data, with each pixel of the object or image, representing the rainfall or temperature in a given area. 2) **Vector format**, whereby objects like roads, buildings, bridges and lakes are represented as overlays or unions of elementary geometric concepts, such as polygons, points, lines ----- (Figure 26 (b); Figure 10) and networks (partitions) shaped by these elements. Due to the huge progress in the production, dissemination, distribution and application of large computerized spatial data (obtained from the objects mentioned above), the acquisition and storage processes of these data leads to the growth of huge spatial databases. As such, we are confronted with huge amounts of increasing spatial data; which means that an end user might face great challenge in understanding these data without the helpful knowledge interpretation from efficient SDs. Hence, SDM is brought under the canopy of DM and is daily attracting more attention. Contrary to classic datasets, S presents varying challenges and so does SDM. Also, because, S not only incorporate spatial relationships within a spatial event, but also includes positional and attribute data, additionally, because the occurrences of spatial events are entrenched in a continual space, sharing a selection of spatial relationships, mining of spatial patterns frequently demands new and improved more efficient techniques. As such, despite the existence of numerous modern high performing systems, the development of new resilient systems is still a huge necessity because spatial data mining is an evolutionary trend. Therefore, we are working towards producing a usable, acceptable, and standard tool for managing and manipulating large spatial data.

1.2 General Hypothesis

For this study, we have set the hypothesis below:

Given:

A set P of S spatial event types {p1, p2… pS}, each $Pi \in P$, *where* $i = 1 \ldots S$, is a vector <

location, spatial event type, instance-id>

Where location assumes a spatial context ***F***,

then

Statement 1: $\forall p = Pi$, if there exists a spatial proximity *SP* over instances in *Pi* and a discrete ***hybrid partition*** of instances in *P*, based on *SP*, a spatial inferential rule could be discovered faster and more efficiently.

Statement 2: Given a spatial index *Si over P*, for dimension $(n) = 2$ in R^n, let i be the instances *Pi of P*, let ***Q*** be any spatial construct:

$$Q = \{q: q = i \in Si\}, if \ q \subset i$$

then

Space complexity will reduce with increase S.

Statement 3: Let dOp be the degree of overlap between instances Pj of P, j= 1...P. Let k be the speed of execution of p, *for* $n > 2$ in ***Statement 2,*** if i increases, then, $k \rightarrow 0$ = true, when $dOp \rightarrow 0$ and false otherwise.

The proof of these hypotheses and outcome of our result showing support for these hypotheses is found in section 6.4.

1.3 BACKGROUND OF STUDY

Owing to the magnanimity of data present in various applications at different locations today, ***further investigation into the measures of accuracy of location prediction is essential***. Precision and recall are two key measures of prediction accuracy for classical DM. Nonetheless, spatial accuracy is not integrated into accuracy measures of predictive models. Investigating on proper measures for location prediction is a big research need presently. With advanced research in **SDM**, the processing of certain predictive or associative models could be explored to further improve their performances. Additionally, further research in SDM in this present age in certain application domains could help us to promptly discover new, useful and relevant insights from large/big data pertaining to any given application domain. Furthermore, according to **Wilson (2002)**, systems characterised by several variables and high levels of interdependence among the elements are typical instances of complex spatial models. These models are often ruled by non-linear procedures and they possess substantial spatial structures. Therefore, the main trouble with building broad complex spatial models is the ability to incorporate the components of a complex system in such a way that they are greatly operational in any precise case. Some general-purpose DM tools are already in place to analyse large commercial databases. However, general-purpose tools for SDM of a complex spatial database needs to be developed also, because hauling out remarkable and beneficial trends from spatial datasets is more complicated than doing same from classical datasets owing to the intricacy of ***spatial relationships, SDTs and spatial autocorrelation.*** **SDM** is used to obviously model the self-supporting behaviour of spatial concentration. By applying spatial data mining technique, one could discover hot spots or small areas representing clusters of a particular kind as related to a given fundamental real-world problem. One could even discover spatial relationships, which will give a meaningful interpretation to some implicit associations between

the variables of any given problem. We could also be able to construct a predictive model, which can be useful in predicting important future outcomes. ***SDM*** is the disclosure of remarkable relationships and features that may exist discreetly in spatial databases (**Ng & Han, 2002**). ***SDM*** can also be seen as the hauling out of knowledge, spatial relationships, or other remarkable trends not overtly stored in SDBs (**Santos and Amaral, 2005)**. An interesting thing about using ***SDM*** techniques is the analytical technique. Spatial data analysis technique could be in the form of (i) **Visualization**: using plots and maps to help the analyst in predictive abilities, (ii) **Statistical models**: using methods that allows for modelling any form of data as a spatial phenomenon and (iii) **Exploratory data analysis**: - using a set of heuristic /assumptions to arrive at a desired result. Spatial phenomenon can be described with the use of n-dimensional objects such as ***polygons***----(Figure 26 (b); Figure 10)***, points, lines or area (Figure 4),*** therefore, the complexity of S, its inherent spatial relationships restricts the practicality of orthodox DM techniques for mining spatial patterns. Finally, **SDM** according to **Samson et al., (2014)** organizes by location what is interesting. Considering the above review there is an impending element of the fact that to carry out a realistic analysis for any ***problem*** in relation to its ***causes.*** This kind of study must be conducted using data, which includes or takes account of the spatial relationship between the observations studied. This is true because there must be a balance in the relationship between ***the place, people, the time and the events*** that would constitute a given study space or case study. As such, for the aim and extent of this work, we have considered all datasets as an organised ***complex system***. Organised complex systems typically exhibit ***the existence of nonlinearities.*** **These are** systems described by several variables, with all elements of the variables having strong interdependencies (**Wilson, 2002)**. *Brains, ecosystems, economies, human beings, cities and languages, all provide good examples of organised complex systems; as such can be modelled as spatial systems.*

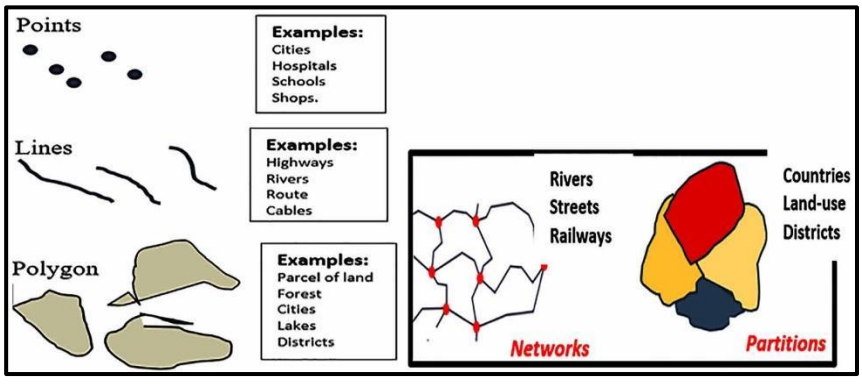


Figure 4: Examples of data types for representing spatial objects

1.3.1 Tasks in mining spatial data sets:

This section is very important, because it introduces aspects of ***SDM*** that every researcher in the ***SDM*** area needs to know. The essence of ***SDM*** is to establish the potential impact of broad ***KDD*** approaches not specially modelled for spatially reckoned data. ***KDD*** for ***SDs*** involves discovering p***atterns, implied regularities, and rules*** hidden in SDs. These activities are gathered into various basic categories based on the underlying knowledge of interest. ***SDM*** involves numerous tasks and, for each task, several diverse methods are often available that could be ***visual***, ***statistical***, ***computational*** or an integration of all of them.

Classification:

By classification (supervised learning), researchers seek to find sets of rules or forms of logic, which determines how to classify an object based on their attribute. ***Classification can be applied in prediction, pattern/trend mining etc.*** Stored information or data are often applied in classification tasks according to **Samson et al., (2014)**, for locating data in prearranged groups. A restaurant chain for example, might extract customer purchases in order to ascertain when customers call, and the frequency of their visits. This information could be used to classify customers into various classes, including ***regular, rare or one-off types of customer***. It can also help to upturn customers' traffic by creating daily specials depending on each group/class of customers.

Clustering:

Clustering (unsupervised learning), is a way of grouping data items according to some logical associations. For instance, we could extract data to identify market segments or consumer attractions. It is a way of putting together objects in a database, which are similar (***clusters***) or putting out all items that are different (***outliers***). ***Clustering techniques can be applied to mining outliers or hotspots in an observation.*** According to Kanagavalli **and Raja (2013)** spatial clustering is often used for the identification of the concentration of certain objects or events in specified locations. Clustering can discover the concentration of particular diseases in certain areas or the spread of epidemics, hence, a root cause analysis will be performed for the recurrent proliferation of disease in that specific region in order to take the necessary corrective measure. Clustering models generally are designed based on optimization criteria, which favours low between-cluster and high within-cluster similarity.

Associations rule mining

Certain association rules can be mined from data. For example, in the customer scenario cited above, we can try relating the sales of a certain good to a certain kind of customer, or determine sales of a good as it relates to the sales of another dissimilar one, if there is any kind of relationship between them. Mining associations, aims to ascertain the uniformities about the elements in large transaction databases by finding all the frequent patterns from transaction data that satisfies the least confidence and the least support (**Agrawal et al., 1993). It was asserted in Boinski and Zakrzewicz (2014) that a*ssociation rule mining can lead to achieving co-location patterns** in a spatial neighbourhood ***or spatial interactions among observations.***

Trend detection:

According to Haimowitz and Kohane (1993, August), a trend is a significant pattern in a sequence of time-ordered data. Therefore, trend detection is the process of finding patterns from time series data. Trends can be multivariate, and can even consist of several distinct phases. Trend detection can be applied to the diagnosis of certain health conditions. Hashavit et al., (2016, July), applied trend detection techniques to online social media streams to predict actual activity increase of social network entities in the context of a search result.

1.3.2 Output patterns

Following the highlight of the different tasks involved in **SDM** as we have discussed above, we now look at possible outcomes from these tasks. Certain simple output design of a spatial process or spatial analysis according to literature, are in the form of

(1) ***Spatial outliers*** (obtained from spatial ***outlier detection)***

(2) ***Predictive models*** (arising from ***spatial classification***)

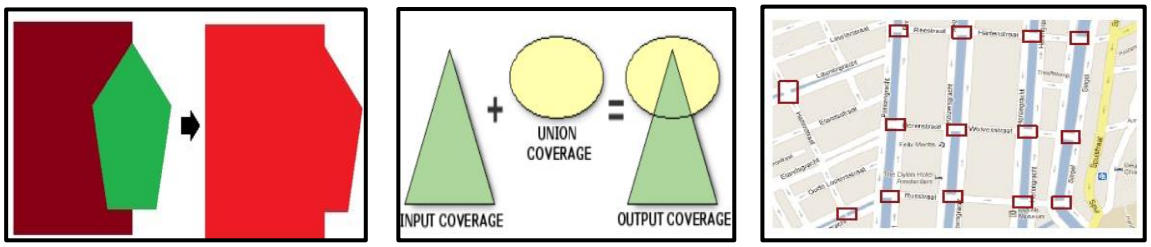
(3) ***Spatial association rule*** (obtained from colocation discovery of spatial datasets)

(4) Spatial clusters (***Shekhar et al., 2003)***.

1.4 Motivation:

Spatial data (S), like very large-scale integration (VLSI), CAD data, geographic (map) data, satellite images or medical images data contain space-related information. S are sometimes denoted as **raster**, which consist of k-dimensional pixels or bitmaps. For instance, a 2D satellite image could be denoted as raster data, with individual pixels registering rainfall in specific areas.

Similarly, S is equally sometimes presented in **vector format as explained in section 1.1**. Where buildings, roads, lakes and bridges are denoted as overlays or unions (computed geometric intersection of two polygon data see Figure 5) of simple geometric concepts, like lines, polygons (Figure 26 (b); Figure 10), points, and the networks and partitions created by these components. Geometric intersection, is one of the operations in a spatial DMGT, it computes the degree of connection between two spatial objects (for instance "river and road network" in a given query location – Figure 5(c)).



(a) (b) (c)

Figure 5: Various types of spatial operation (a) and (b) union of two (2) maps. (c) Showing intersection of rivers and paths (road network)

SDM has pulled a great deal of interest from not only the spatial information industry but also the whole society in recent years, as a result of the broad obtainability of enormous volumes of S and the impending necessity to convert such S into an expedient spatial understanding and information. The understanding and information gained (spatial) is useful for applications going from providing the public service information as regards the location of electric cables and telephone wires, forestry, sewage systems and even ecology planning. A Spatial data repository design having already materialised is the **spatial data warehouse (SDW)**. This is a warehouse of several unrelated data sources, structured under an integrated schema on a single node (data site) in order to ease management decision making. The tools for such repository systems includes S cleaning, S incorporation, and **spatial on-line analytical processing** (SOLAP). These technologies possess analytical mechanisms with functionalities of *consolidation*, *summarization*, and *aggregation* as well as views.

Data rich but information poor paradigm, describes a situation where there abides huge availability of data, but with a corresponding huge need for the efficient and powerful data analysis tools for the management of these data for information acquisition, especially in the spatial data field. The ever evolving, large volume of S, gathered and saved in huge and several S depositories, have extremely surpassed human capability for understanding, because efficient technologies are lacking. Thus, S collected and stored in repositories turns out to be "spatial data tombs ". This

spreading gap between S and spatial information demands for a methodical development of *SDM* that for converting "data tombs" to "golden nuggets" of information. However, an imminent question is; "What kind of DM methods should be operated on spatial data sets?" what tools are more useful for the discovering, the spatial associations between a set of spatial objects. These tools are expected to offer the opportunity for proper ordering, to ascertain subclasses of objects that are spatially associated. Moreover, during a mining process, noise and clusters also need to be recognized by spatial cluster analysis, thus, spatial classification should be provided to build models for prediction that are depending on the significant set of attributes of the spatial objects.

Therefore, the crucial challenge in SDM is the efficiency of SDM algorithms and the effective application of SDM technologies, owing to the enormous amount of S, and the complexity of SDT and spatial methods. Although SOLAP devices and mechanisms assists multi-dimensional analysis and decision-making. Further, S analysis devices are necessary for in-depth analysis, such as spatial data classification, spatial co-location mining, and spatial outlier detection. In addition, vast volumes of S can be accumulated beyond SDs and S warehouses. Therefore, analysing these spatial data in such different forms effectively and efficiently becomes a challenge. These challenges form the basic motivation for this project.

1.5 Project Justification:

Obviously, the ever increasing and wide accessibility of data assembled in the form of paths (lines or routes) has led to progress in research relating to behavioural aspects of supervised themes including people, vehicles, and wild animals. By means of these kinds of data (trajectory) collected through strategies, such as mobile devices, RFID and GPS, complex design and complex queries can be modelled to pick significant data depending on precise events of concern **(**Moussalli et al., 2014; Moussalli et al., 2015). Regardless of the remarkable effort in SDM research methodologies, current researches have shown that sensitive areas and issues of SDM needing attention are still in existence. In Kamlesh et al., (2015); Meng et al., (2014); Mauder et al., (2015); Billings (2013); Kang et al., (2014); Moussalli et al., (2014;2015); Huang and He (2015;2014); Secchi, et al., (2015); Tian and Scholer (2015) and Eldawy et al., (2015), these problems include location privacy, inconsistency, scalability, and uncertainty in handling S. These problems emanate from Physical causes causing inaccuracy and noise in ***location-time*** data, as such, data analysts constantly search for solutions to them.

Chapter 2 examines the basics of SDs and demonstrates their fundamental architecture, and components. The chapter concentrates on the query Language, data models, indexes, query

processing, and query optimization of a SD that allows SDs to be an essential instrument for data storage and retrieval in high dimensional spaces and multidimensional datasets.

Chapter 3 is a thorough description of the framework that underpins spatial analysis and spatial modelling. The chapter explains the general context/process adopted for controlling, planning, and structuring this research work. The chapter describes the specialised technique employed in searching for scientific truth, creating correct elucidations of real-world phenomena, and building efficient systems and presents an innovative method in which variables from a given dataset are extracted in the form of spatial attributes/variables, if the object can be interpreted or represented as an *n-dimensional Euclidean* space (R^n).

Chapter 4, introduces new methods that improve the efficacy of some current existing spatial models/algorithms. Our incentive here is to enhance the performance of these algorithms w.r.t their techniques for interpreting and analysing large databases. The proposed systems include, (1) **Xdbscan: A Proposed Algorithm for Improved Clustering** section 4.1 **and** (2) **Ax-Tree: An Index Structure for Large Spatial Datasets** section 4.2. This chapter elucidates on the design, implementation, experiment and evaluation of these models by giving a thorough overview of how the proposed models work through the two algorithms above.

Chapter 5, In this chapter, all our finished/published work is presented. We also introduced some projects that integrate all the methods described in chapters 4. This chapter actually focused on ways of improving existing systems and proposing novel techniques for mining data from spatial databases effectively. In section ***5.2***, we carried out a detailed investigation on the background and basis SDs in the topic **"Spatial databases - an overview".** This project covers an introduction to the SD subject matter. In section ***5.3***, we discuss the topic "**paX-DBSCAN: a proposed algorithm for improved clustering**". The project is a proposed model for parallel clustering of large SD systems that focuses mainly on the application of parallel computing methods and techniques for the bulk loading of the existing X-tree. We present the discussion of another project that was accomplished, on this programme in section ***5.4.*** **"Large spatial datasets: Present Challenges, future opportunities".** The project is a thorough investigation into the impending problems and available opportunities for large (spatial/non-spatial database) systems. Section ***5.5*** describes a new and improved algorithm for ***DBSCAN*** spatial clustering method. A *conference paper* titled **"Spatial Clustering in Large Databases Using Packed X-tree".** Section ***5.6***, is an aggregate classic that produces an efficient technique for spatial modelling of human skin related pixels. This project combines all the work done during the course of this research. We applied the model to detection human faces in an image; the result is presented in Section ***5.7***. The project described in Section ***5.8*** **"Large Spatial Database Indexing with aX-tree",** discusses our main work in this research,

geared towards an improved structure for indexing large and high dimensional datasets using spatial data models.

Chapter 6, presents the result of our work, including the implementation of the proposed algorithm, outcome and evaluation. The chapter also iterates the fact that; after the analyses of the existing methods; we are of the opinion that new efficient methods are still needed.

CHAPTER 2. OVERVIEW OF SPATIAL DATA BASES AND SPATIAL DATA MINING:

Spatial databases (SDs) preserve space information, appropriate for applications that require monitoring the site of events and objects in space. SDs describes the ultimate picture of the object of a dataset that emanates from geographic, spatial or geometric objects. A spatial database system (SDBS) is a database system (DS) that deals with spatial and/or temporal DTs in their query language or data model (***Samson et al., 2017)***. The attributes for referencing the spatial or geographic quantities of an object in SDs, allows the objects assume a two (2) or three (3) dimensional position in space. SDs have wide applications in geographical information systems (GIS), including remote sensing, medical imaging, geo-marketing, image databases, and several other systems where S are utilized. A major difficulty to spatial data mining (SDM) is the critical examination of capable SDM methodologies, which results from very big size of spatial data (S), and the complications of spatial data types (SDTs) and spatial access methods. The final objective of this project is to develop some novel theoretical concepts and data mining (DM) systems. This chapter examines the basics of SDs and demonstrates their fundamental architecture, and components. The work concentrates on the query Language, data models, indexes, query processing, and query optimization of a SD that qualifies SDs as an essential instrument for data storage and retrieval in high dimensional spaces multidimensional data. More about SDM is found in **Samson et al., (2016); Samson and Lu (2016)**; **Samson et al., (2017); Samson et al., (2018); Samson and Lu (2018)**.

2.1 Background in SDM

The widespread and ever-growing accessibility to data gathered from geographical information systems (GIS) equipment and technologies has made it extremely hard to maintain these data using current SD techniques. Consequently, has led to progresses in research relating to the operational characteristics of supervised elements. GISs can manage all the main responsibilities of information extraction (including data verification and data input, output and presentation, storage and manipulation, data conversion and communications with the end users) efficiently, from massive datasets. This indicates that GIS are broad DBMS with the ability to manage all the above-mentioned responsibilities (Rigaux et al., 2003). Nevertheless, depending on the underlying system, the elements in these (GIS) data must be characterized correctly, for real-world data interpretation (that is interpreting real-world data in order to produce a real pattern) in order to

improve the efficiency of the performance of the database. In order to realize this database objective, data analysts frequently search for suitable data models that can save the data elements in the database efficiently allowing for a better DBM. SDs preserve space information, fit for systems that require monitoring the site of events and objects in space. SDs describes the ultimate picture of the elements of a dataset emanating from geographic, spatial or geometric objects. A SDBS is a DS that deals with spatial and/or temporal DTs in their data models and query languages. Different from the relational database, SDs query data not only on their features alone, rather, they in addition, possess the capabilities of performing queries on data elements concerning their localities. SDs are designed to supplement the relational database, based on some structured system architecture (see Figure 6 and Figure 7).

This chapter examines the basics of SDs describing their fundamental operations architecture, and components. Figure 6 shows a typical database environment.

2.2 Architecture for SDMS

SDBS, according to Ester et al., (1999), are relational or classical databases with an additional notion of spatial extension and spatial location. The obvious locality and expansion of elements define inherent associations of a spatial neighbourhood. The efficacy of many KDD algorithms for SDBS (Ester et al., 1997) relies mainly on an effective handling of the underlying neighbourhood relations because in a single execution of a KDD algorithm, the neighbours of many objects/elements have to be examined. DBMS architectures are usually engaged in the creation of SDs. However, whilst classical/traditional databases understand several figures and characters DTs (Shekhar, 1999), supplementary functionalities are needed for them to process most SDTs. These functionalities are usually called features or geometry. Three fundamental architectures for designing SDMs according to Güting (1994) as shown in Figure 7 are the layered, dual and the integrated architectures. In the layered architecture, databases use the standard DBMS built on top of the databases, while a spatial tool is set as the top layer. For the twin architecture, the layer on the top is the layer for integration, which will incorporate the standard DBMS and spatial sub-process in the base layer. SDs defines the important picture of the dataset elements that emanate from spatial or geographic objects. Below we present the primary aspects of a SDs that are modelled

1) ***Where (that is the*** spatial components***)***
2) The ***what (that is the*** attributes***).***

The two aforementioned factors determine the spatial and attribute data that constitutes the spatial database. While spatial objects define the locality of the object of interest, attribute data specifies features and actions at that location ***(i.e., when, how, what, how much and so on).*** Nevertheless, presenting these data in a form that is understandable to the computer, calls for clustering the data into layers depending on single components with related features ***(for instance, one layer could be elevation, another layer waterlines, topography or temperature)***. Notwithstanding, the properties of each data layer (including accuracy, scale, resolution and projection) has to be established by picking suitable features for individual layers. At this point, the logical layer of the SDBMS comes in. According to Rigaux et al., (2003), the logical layer (LL) holds the definition of the schema of the SD, which defines the organisation of the information that is managed by the system. The LL also holds the rules that should be satisfied by the data in the database. Describing the schema will allow for additional database operations (data insertion, deletion etc.) in addition, it permits database query by using a proper spatial query language. Conversely, it is right to assert that the specific constraints, operations and structures delivered by the SDBMS rely absolutely on the logical data models, supported by this SDBMS according to Rigaux et al., (2003).

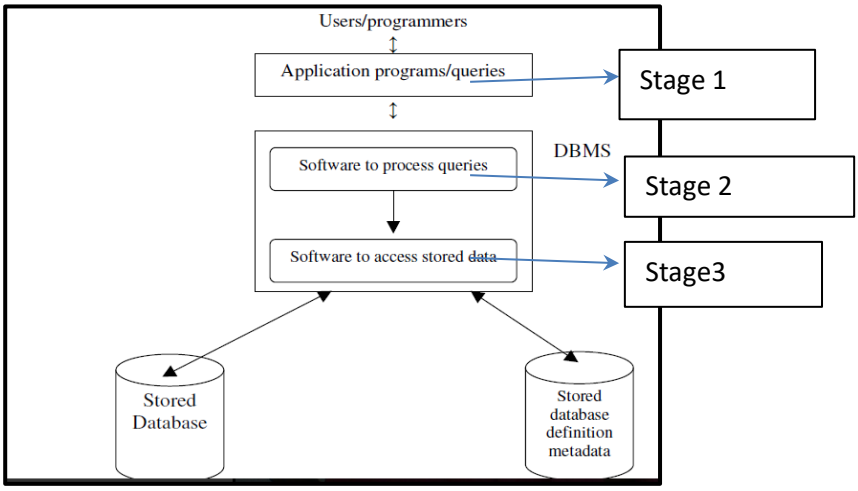


Figure 6: A typical environment of DBM

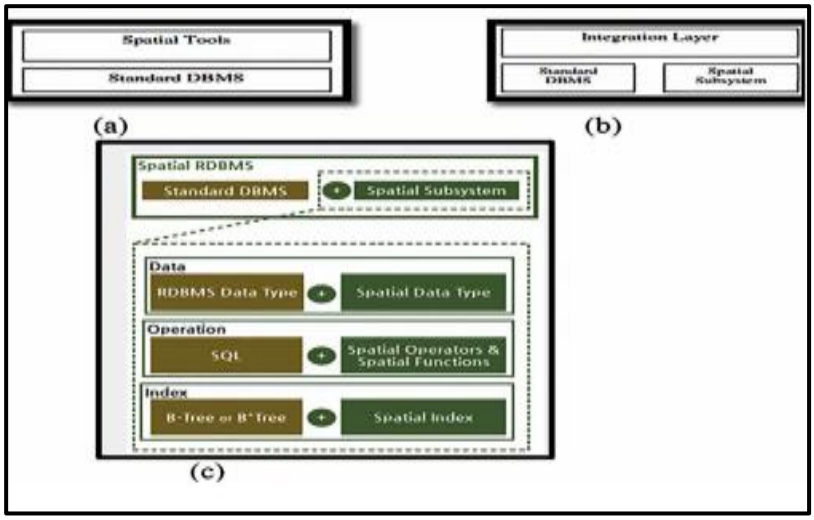


Figure 7: The architecture of SD illustration (a) layered (b) dual (c) integrated (Güting, 1994)

Regardless of all the architectures discussed above, the current difficulties confronting the handling of big spatial database (Eldawy and Mokbel (2015, April); Li et al., (2015); Alkathiri et al., (2016) and Economides et al., (2013)), has necessitated the invention of new methods, techniques and software tools, as well as parallel computing hardware architectures, that meets the constraint of timely and effectual management of very large data. Owing to the steady growth in the size of S effortlessly available from various site-based systems, there has been an outburst in the capacity needs of computer storage and the control units. This has led to the advent of the notion of parallel programming. Numerous architectures have been proposed and several frameworks have similarly been developed to benefit from the advantage of recent hardware developments. These frameworks (distributed data models with the ability of accessing many storage and processing units is used) and architectures include: Hadoop Framework for distributed computing (HFDC), Network Computing, Cloud computing platforms, and CUDA Computing (which uses an array of processors in GPUs manufactured by NVIDIA). Currently, OpenMP programming model based on Intel Xeon Phi coprocessors (Alkathiri et al., 2016) has also evolved. The Hadoop framework distributed computing has demonstrated very useful in areas like Spatio-temporal data analysis, Biomedical Imagery analysis, Geospatial data analysis and even countless other systems that involve large sizes of data. The work in Economides et al., (2013) presents a new SD Design that is a Hadoop based framework for efficient complex spatial query management with high performance. This framework adopts the R-tree spatial indexing structure for managing SDs within the Hadoop framework, to additionally enhance the performance of the structure. Basically, two (2) major parts that sets up the distributed architecture for manipulating massive volume datasets, including; (1) distributed file systems-DFS (for connected nodes that builds a cluster, for data storage – e.g. *Hadoop*) and (2) a programming model/module for processing the

dataset, which two major functions comprise mapping and reducing – example *Map-Reduce*. The mapper in this instance accepts an incoming data (input) key-value pair from an input reader. More about recent state-of-the-art architecture for spatial data mining can be found in **Samson et al., (2016); Samson and Lu (2016)**; **Samson et al., (2017); Samson et al., (2018); Samson and Lu (2018)**. Furthermore, SDs use spatial index to hasten database operations. SD architectures use standard DBMS. In carrying out this research work, we applied the layered DBMS model in handling spatial datasets. We used predictions instead of observations, as a collection of learning examples to which we applied the automated learning tools. We also used simulations to grow a bigger set of learning examples other than observations, owing to the fact that they can vary a larger total of factors over a wider range. For example, in some real-world cases, effects of the causal variables may be difficult to be tested on real objects by observations because of the possibility of the presence of harmful material.

2.3 Application areas of SDM methodologies

In Kaundinya et al., (2013), the application of SDM technique was proposed, for effectively picking the localities and ascertaining the installed capabilities for establishing dispersed biomass power generation systems from the perspective of distributed energy planning for rural regions. **Kanagavalli and Raja (2013)** suggested the application of SDM for efficient management of the challenges confronted by outer rural areas, in analysing and forecasting market scenarios and better manage their economy. In **Li et al., (2012)**, spatial DM techniques is applied to mining meaningful and useful information from e-government Information System, **Li and wang (2005)** applied SDM to **(a)** remote sensing classification, **(b)** landslide-monitoring DM, and **(c)** uncertain reasoning. **Wu et al., (2013)**, used a multisource spatial DM approach for climate simulation, they applied this method to the land cover mapping of farming-pastoral ecotone of North China. **Ramos et al., (2015)** applied spatial data mining methodologies to the improvement of the prediction of drugs demand. **Chen et al., (2015)** applied the SDM method in parallel and distributed spatial outlier mining in a grid environment. **Dawei et al., (2013)** applied SDM techniques to Crime hotspot mapping using the crime related factors. **Fan (2014)** applied SDM techniques in predicting the best method for selecting appropriate sites for situating emergency response centres. **Kim et al., (2014; 2012)** proposed a new framework of spatial co-location pattern mining for pervasive GIS. In **Kawulok et al., (2014),** a spatial based system for skin detection was presented. The system applies a discriminative feature space as a ground for spatial analysis of skin pixels, based on textural features extracted from a skin colour probability map. Interestingly, recent advances in

knowledge discovery with SDM methodologies has seen this field of study delving into very complex real-world scenarios for example, moving object databases have enjoyed all manner and dimensions of spatial data mining technology in the process of determining interesting patterns, see Table 1. More about recent state-of-the-art application areas of spatial DM methodologies can be found in **Samson et al., (2016); Samson and Lu (2016)**; **Samson et al., (2017); Samson et al., (2018); Samson and Lu (2018)**.

Table 1: Advances in spatial data mining methodologies

Area	Sources
Mining knowledge from moving object	Brakatsoulas, et al., (2004); Guo and Cui (2008); Bogorny et al., (2008); Lee et al., (2009); Li et al., (2010); Han et al., (2010); Hajari and Hakimpour (2014); Urbano and Cagnacci (2014); Vieira et al; (2009); Romero, A. O. C. (2011)
Hierarchical modelling of large spatial dataset	Banerjee et al (2014), Reza Najafi and Moradkhani (2013), MacNab et al (2014), Chen and Brown, S. D. (2014), Shirk et al (2014), Kang et al (2015), Daniel et al (2015)
Application software for analysing spatial data	Fowler and Bridge (2015),
Grid – Based spatial data mining/ Parallel – Distributed Spatial Data Mining	Lakshmanan, V. (2012), Chen et al (2015), Teegavarapu et al, (2012; 2011), Heinecke and Pflüger (2013), Samarah (2012), Fan and Luo (2009),
Polygon based spatial data mining	Wang and Eick (2014), Joshi et al., (2014).
Modelling uncertainty	
Managing scalability in spatial data	

2.4 Techniques used for spatial data mining

Li and wang (2005) introduced three **(3)** new SDM and knowledge discovery-based techniques including **(1)** image classification by integrating Bayesian classification and spatial inductive learning from GIS database, **(2)** Cloud model, which integrates fuzziness and randomness, and **(3)** data field that emits the energy of experimental data to the universal dialogue. **Li et al., (2012)** applied clustering technique of SDM based on the *EM* **algorithm**. The algorithm is a typical spatial clustering algorithm that employs the probability method in distributing each data to their determined classification (group) rather than calculating directly by the distance. **Kaundinya et al., (2013),** applied a *k*-medoid clustering algorithm to split a given region into clusters of villages;

afterwards the system locates the biomass power generation systems, as the medoids. The best *k* value is selected (iteratively) by executing the algorithm for the complete search space, using varying values of *k*, alongside the demand–supply equivalence constraints. The best value is then chosen such that it diminishes the overall cost of system installation, transportation of biomass, and transmission and distribution.

Wu et al., (2013), using the C4.5 algorithm applied the inference rule of SDM to separate forest types depending on the regular grids in the data of the International Geosphere-Biosphere Programme Data and Information System. **Ramos et al., (2015)** used the generalized linear models (GLM) algorithms and support vector machines (SVM) with linear and Gaussian kernels respectively.

The system was developed using Oracle DM. **Chen et al., (2015)** proposed an innovative parallel & distributed spatial outlier mining algorithm (PD-SOM) which is recommended to simultaneously detect global and local outliers in a grid environment. **Dawei et al., (2013)** introduced a new crime hotspot-mapping tool Hotspot Optimization Tool (HOT). HOT is an application of SDM to the field of hotspot mapping by capturing the dissimilarities between two (2) groups in a spatial dataset. **Fan (2014)** used a hybrid analytical method spatial data association mining method that identifies the correlation rules that hold some emergency information and geographical factors. Apart from the techniques above, the SQL is now the top standard for data querying. Likewise, it is used for expressing most spatial queries. Due to structural differences, it is often very difficult to write effectual SQL based queries in relational and spatial data programming models Ajiand and Wang (2016). Spatial query processing is the primary functioning element to support spatial applications according to Zhong et al., (2012, May). Nevertheless, cutting-edge methodologies of spatial query processing (SQP) are facing major difficulties as the data grows and user accesses increase. Having said all that, the need to frequently analyse massive volumes of S stored in a spatial data warehouse demands querying the data warehouse store using spatial online analytical processing ***(SOLAP)*** systems. Rao et al., (2003, November) presented an effective searching technique that uses the base-language "Consistent" for finding all the leaf nodes (that is the actual data objects as referred to by an index) that are stable with the base-language (query predicate). Additionally, a novel state "consistent" predicate was also introduced called "Partial-true" which works with a new search algorithm for effective results during an online analytical processing (OLAP) query connecting a spatial data warehouse. More about recent current techniques can be found in **Samson et al., (2016); Samson and Lu (2016)**; **Samson et al., (2017)**.

2.5 Process of spatial data mining:

Researchers have tried to highlight the major processes or stages that must be accomplished in order to complete an SDM task. In **Li et al., (2012),** the process of SDM must include the following

i) Problem/system investigation

ii) Item selection

iii) Element pre-processing/cleaning and conversion

iv) Knowledge representation, mining, extraction and accessing.

In Figure 8, we have summarised the processes/procedures involved in knowledge discovery through spatial data mining into six (6) crucial stages.

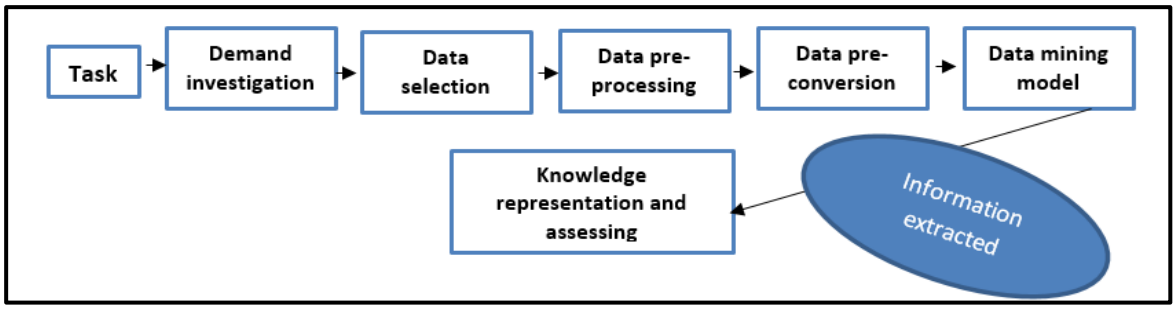


Figure 8: Typical spatial data mining process

2.6 Presenting result from spatial data mining process:

The result of every DM task and assignment must be presented in a very clear and understandable form that can help organizations, economies or individuals easily get the meaning of the entire process. In **Li et al., (2012)** various kinds and nature of data could be found in most large (big) datasets this includes the following: Attribute data Raster and Vector data, and Statistical data. Based on these kinds of data according to the authors**,** the result of a SDM task can be presented in form of ***visuals*** (e.g. statistical graphics like ***trees----Figure 9***), ***charts***, ***histograms***, ***pyramid diagram,*** and **statistical analysis**

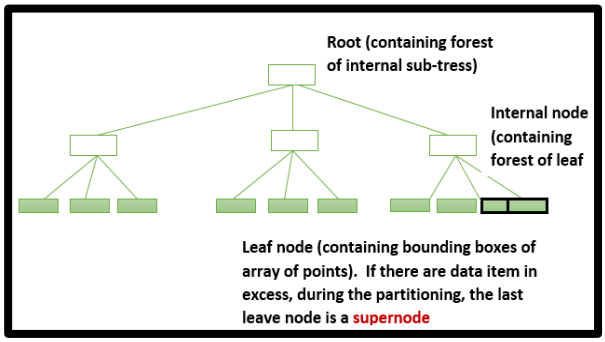


Figure 9: Typical example of a tree structure

2.7 Present challenges of SDM

Over the years, SDM has advanced through the numerous computational difficulties and challenges and there has been tremendous improvement and advancement to this interesting field of study. However, reports from Samson et al., (2016) on current literature as regards this field, indicates that there are still areas demanding serious advanced research. Large spatial datasets according to Economides et al., (2013) have the capacity to accumulate very comprehensive information, including multi-variable data sets, that results in a huge storage and processing demand. The main challenge confronting large SDs is the difficulty in utilization of storage and computational complexity. Inopportunely, latest works (Eldawy and Mokbel (2015); Li et al., (2015); Alkathiri et al., (2016); Ravada, S. (2014); Economides et al., (2013); Bhosale and Gadekar (2014); Sagiroglu and Sinanc (2013, May)), have discovered that the critical necessity for managing and analysing big S is stalled by the absence of dedicated systems, algorithms, methods and techniques that supports such data. Tackling the challenges related to big spatial systems translates to the management of streams of semi-structured, structured, and unstructured data in dissimilar databases either on-ground or in the cloud. Li et al., (2015) similarly, added that the challenges confronting the processing and management of large amounts of S is intensified by the incessant increase of S as regards scope, depth and scale, all of which had rendered current systems for manipulating spatially referenced objects inefficient, insufficient and inadequate. In Alkathiri et al., (2016), it was stated that large capacities of data gathered in various formats, with large difficulties and constant production, have postured an overwhelming problem in the business and the +scientific world alike thus, mechanisms, methods and hardware existing for a decade ago, have encountered great restrictions in managing such huge data. Ravada (2014) acknowledged a number of challenges that face the management of enormous spatial datasets in the notion of managing big data. These challenges include

1) **Time complexity**:

Under time complexity, two major issues are visible, these challenges are listed below:

- **a.** This is the measurement of time taken to make valuable data available for analysing the discovery from spatial and temporal relationships between diverse spatial data points (objects).
- **b.** |This is measurement of the time taken for data loading in order to allow information or data accessible for usage.

Other challenges faced by spatial data mining include:

2) The creation of suitable spatial indexes to aid processing productivity.

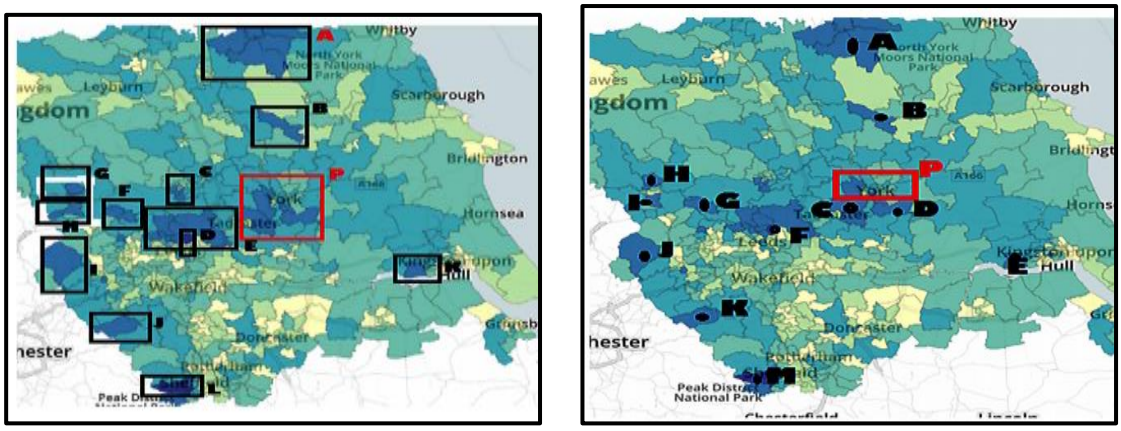
4) The ability to access code from SD programs, built during quite a number of years.

5) The ability to design predictive analytics for several systems.

The challenge of BSD, majorly originates from the *complexity (variability), size (volume),* and *rate of growth (velocity)* according to Bhosale and Gadekar (2014), which makes them complex in terms of collection, management, processing or analysis, by current technologies and tools. In essence, Sagiroglu and Sinanc (2013, May) reported that combating all these challenges requires inventive procedures for data analysis, categorised by its three main components: velocity, variety, and volume. Ajiand and Wang (2016) gave a brief overview of the major difficulties confronting big spatial datasets in contemporary times, these includes a) large volumes of multi-dimensional data, b) high computational complexity and c) complex spatial queries. Further to the challenges highlighted above are more imminent ones which are even more common in the SDM discipline.

2.7.1 Mining in Spatial Object-Oriented Databases:

The concern here is how an object-oriented programming (OOP) approach can be adopted to construct a SD and how knowledge can be extracted from these databases. The question is key because several scientists or researchers have revealed that OOP databases may be an enhanced option for managing spatial data instead of the traditional classical relational or extended relational models. For instance, rectangles (Figure 10 and Figure 26) more complex (spatial) objects and polygons can be simulated naturally in object-oriented programming (OOP) databases.



(a) (b)

Figure 10: Typical spatial data object (a) How to enclose the geometry of the spatial object (towns around Yorkshire County in this case) using a rectangle (MBR) (b) Example query point/location (point **P** in red colour)

2.7.2 Parallel DM:

Owing to the huge size of S used during several computations, there is this challenge of employing parallel machines or circulated farmhouses of workstations for mining purposes. This means that researchers are faced with the task of producing a system for parallel knowledge discovery, which is able to accelerate computation procedures significantly. Thus, we expect that there will be an emergent research issue both for the relational or classic DM and for SDM as it concerns parallel processing.

2.7.3 Other Clustering Techniques:

Another remarkable futuristic track in the area SDM is the collecting of possible overlapping objects like polygons or shapes, as opposed to the collecting of points (data elements represented as n-dimensional points). Clusters (or groups of similar objects) can also reserve supplementary information/data about the objects they contain (for instance, degree of membership). As such, an ambiguous clustering technique could be used to gather objects that have the same distance from the mean or medoid (clusters of datasets or their representative objects with minimal average dissimilarity).

Other challenges that need further exploration and research include:

- **Mining information from moving object**
- **Modelling uncertainty in spatial data**
- **Modelling knowledge from classical/traditional databases through SDM methodologies.**

2.8 Autocorrelation:

Failure to control autocorrelation according to Jerrett et al., (2003) can give rise to high false positives in tests of significance, which may indicate prejudice emanating from missing variables or a collection of variables. Spatial analysis has been applied (by many researchers) for handling issues of autocorrelation. In **Fuller (2012**), spatial autocorrelation is an effective way to systematically ascertain spatial trends of inequalities between variables. Spatial autocorrelation according to **Legendre and Fortin (1989)** often occurs in ecological data, and several ecological theories, methods and models indirectly assume an underlying spatial trend in the dispersals of organisms and their environments. Autocorrelation originates from the fact that objects of a certain population or community (including the geographic/social environment as a whole), which are

close to one another in time or space are most likely to be affected by the same generating process. By this definition, **Chen et al., (2011)** indicated that spatial autocorrelation shows high correlation between a data item/object or a variable with itself in space. **Legendre (1993); Rossi and Queneherve (1998)** established that spatial autocorrelation quantifies the match/connections between data samples for a given element as a function of an underlying distance between data items in the sample. In **Dale and Fortin (2009) spatial autocorrelation is seen to** simply portray self-dependence of S. Meaning that the discrete observations obtained from a chosen sample will always include information present in other observations, thus the effective sample size (say *k*), is expected to be less than the total amount of observations (**N**). This dependency as suggested by the authors, presents a huge challenge that affects the performance rate of statistical significance tests when it is positive and therefore has to be adjusted in order to yield an improved measurement of goodness-of-fit. **Autocorrelation is seen generally as the measure of the similarity/relationship or interdependence of an object in space with surrounding objects**. Spatial autocorrelation according to **Noel et al., (2013)** is intuitively necessary in geographic space, since without it, the distribution of phenomena would be independent of location, and thus random. Clearly, the spatial distribution of phenomena is not random. As Tobler's First Law of Geography denotes, things near or close to each other are more likely to be alike than those further apart. For example, in a city segregated by religion such as Belfast, a Catholic's neighbour is more likely to be another Catholic than a Protestant. The fact that many geographic phenomena are **autocorrelated**—being same or similar to its neighbours—provides a problem for classical statistics that assume independence of observations in its sampling and analysis. Consequently, a number of statistical techniques have been developed to measure and model patterns of spatial dependency in data, which include the variogram and methods of fractal analysis (**Noel et al., 2013).**

2.9 About spatial and temporal DM

Spatiotemporal databases are designed to handle objects or data with geometries changing over certain periods. Typical applications, which could generate this type of data, might include geographical systems, transportation systems, surveillance systems, environmental systems, mobile communication systems, and so on. According to **Jia-Dong et al., (2003),** these types of datasets have some distinctive features that distinguish them from classical and transactional datasets. These differences mean that changes could be ***continuous, in contrast to traditional databases, where it is assumed that data changes over an unambiguous update.*** To evade

continuous database updates on a spatiotemporal database, the explanation of the changes, which is a function of time, has to be stored. Similarly, novel concepts, techniques and methods are constantly required for the extraction of more thorough and comprehensive information from the massive storehouses of spatiotemporal data that are accruing. Methods and techniques for identifying spatial and temporal trends or patterns over numerous datasets will improve the capability to transform S and generate useful information, ***such as causal relations and trends***. Spatiotemporal data is usually modelled by augmenting temporal databases or SDs. That is, spatiotemporal data is demonstrated in two ways:

(1) By adding spatial characteristics and operations in temporal databases,

(2) By adding temporal characteristics and operations in SDs.

In conventional/classical/traditional databases, attributes that contain temporal/spatial information that are being operated solely through application programs, with little help from a DBMS. A spatiotemporal database is a sort of database, which supports aspects of both space and time. SDBS deals with spatial and/or temporal DTs in their query language and data model. They normally support SDTs in their execution, presenting at the minimum spatial indexing and effective procedures for spatial join.

2.10 Features of S

Sherkar et al., (2005) concentrated on the distinctive features that differentiates SDM from classical DM and was able to categorise them based on the following four classifications: **statistical foundation, data input, computational process, and output patterns**. Consideration of spatial predicates becomes imminent when querying spatial databases; this is because SDs are sometimes faced with the presence of constraint (such as topological constraint) that makes current classical database solutions inappropriate **(Clementini et al., 1994).**

2.10.1 Input

The data inputs of SDM as we have discussed earlier are additionally composite than the inputs of relational DM because they comprise of protracted objects/elements like **polygons**---- (Figure 26 (b); Figure 10)) **lines**, and **points**. The data inputs of SDM have two distinctive types of factors: spatial factors and spatial non- factors.

2.10.2 Fundamentals and statistical background of background

Spatial statistics emanates if the data to be examined are some points objects/data in Euclidean space. According to **Chang (2004),** spatial data resides within a distance, in an n-dimension space usually represented by R^n. Data that could be measured on some surfaces consisting locations of points, could likewise generate statistical analysis. ***Statistical analysis of S might include either discovering the possible elements present in the S, or creating the constraints for an intended model.*** Likely features; e.g. spatial outliers, co-location of elements, spatial patterns/trends and even spatial relationships, could be revealed using spatial statistics.

2.10.3 Processes of computational

Adhikary (1996) purported that two main types of spatial operations exist that could be executed on a spatial dataset; **map overlay** and **spatial join.** Spatial join could be implemented using the R-tree structure, x-tree structure or any other hierarchical structures as proposed by **Brinkhoff et al., (1993)** nevertheless, map overlay (see section 1.4, figure 5) encompasses the mixture of the qualities and features of two or more data layers on a data frames (spatial map) layout to yield a desirable output. As identified by **(Güting, 1994)** other operations can be classified or categorised into four main categories (based on the type or characteristics of the data input), all these categories include spatial operations on

1) Sets of objects, e.g. **Closest, sum, near…**

2) Continuous figures, e.g. **Distance, area, region, perimeter …**

3) Functions returning atomic SDT, e.g. **Minus, Intersection, contour, plus….**

4) Expressive relationships, including **adjacent, intersect, contain…**

2.10.4 Spatial Data Types

Unlike regular relational or classical DBMS, that saves data as figures, alpha-numeric, symbols or alphabets. SDs save abstract ADTs as polygons (Figure 11 (a) and (b)), coordinates (latitude coordinates and longitude coordinates defining a particular position on earth's surface), lines, topology or other DTs (see figure 10) that can be mapped according to (Ernest and Djaoen, 2015; Samson et al., 2013). The description and application of SDTs is the most vital matter in the design and construction of a SDBS (Güting and Schneider, 1993). According to Schneider (1999), SDTs are compulsory in the modelling of geometries and in the proper representation of geometric data in a SDS. According to the author, the basic/simple DTs consist of line, points, and region (areas). The composite types are partitions and graphs (such as networks including roads, rivers etc.).

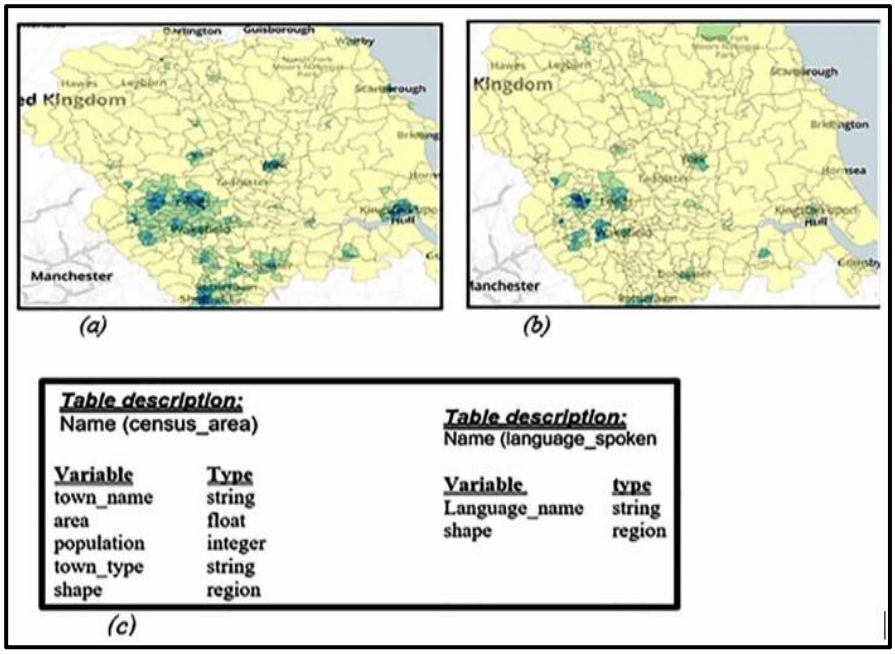


Figure 11: Sample dataset (spatial) (a) and (b) thematic maps of a county, showing places that census was carried out and places where English language is major (c) relational tables representing the two spatial objects (a and b)

Table 2: How to represent a single record, meant to be stored as an item in a new database table

Town_id	Name	Population	Area	Boundary
132	Huddersfield	120000	1	((0,0),(0,1),(1,0),(1,1))
....	.....			

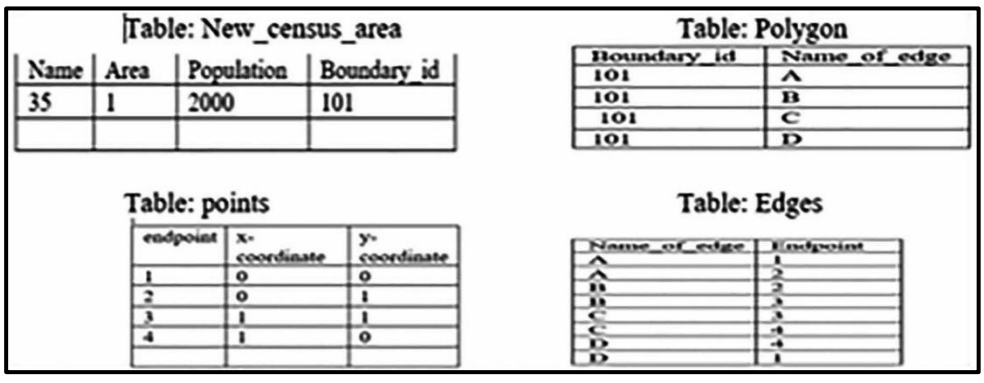


Figure 12: Example relational or database tables that stores the spatial properties of the object

In Güting and Schneider (1993) SDT including regions----(Figure 26 (b); Figure 10), lines (paths) and points are defined as qualities of a spatial objects, which defines the objects characteristics

irrespective of whether the DBMs uses a composite object, relational model, OO models or any other data paradigm. SDs consists of collection of both spatial and non-spatial data that is enhanced to efficiently store and investigate data items situated spatially. Spatial databases offer additional functionalities, (compared to relational ones that work only with character, calendar or numeric data), which allows processing SDTs (Velicanu & Olaru, 2010). SDTs that is, data types that generally describe the actual location, feature and shape of geometric objects according to (Ernest & Djaoen, 2015) are categorised into 2: geography and DTs. For geometry data, they can be saved using their x and y coordinate values, by this mechanism, the x and y coordinates consequently, places the spatial object (lines, polygon/region------ (Figure 26 (b); Figure 10)) or points) on a 2-dimensional Euclidean space. Geography types of S, according to Ernest and Djaoen (2015) store data, based on spherical earth coordinate systems. Therefore, the spatial object is saved using their latitude and longitude coordinate values. More explanations on SDTs can be seen in Samson et al., (2013) and Samson et al., (2014), for a basic instance, we shall see a demonstration below.

Supposing the underlying issue is to find the closest town to the centroid of a certain place (for instance Yorkshire Counties marked Pin Figure 10 (b)), We represent the cities A--- M (from section 2.7.1---- figure 10) in our database. Therefore, we save the cities in the form of points, by taking the coordinates values of x and y. This helps us create a new table called New_ census area (shown in figure 12). Figure 11 portrays diverse types of spatial objects and how they are saved on a relational table (Figure 12) and Figure 13 is an indication of a broad summary of the various SDT and their definitions.

Geometric Type	Representation	Description
point	(x, y)	Point in space
line	$((x_1, y_1), (x_2, y_2))$	An infinite straight line defined by two points
lseg	$((x_1, y_1), (x_2, y_2))$	A line segment defined by its two endpoints
box	$((x_1, y_1), (x_2, y_2))$	A rectangle (rectangular box)
path	$((x_1, y_1), \ldots)$	A *closed* polyline (closed path)
path	$[(x_1, y_1), \ldots]$	A polyline (open path)
polygon	$((x_1, y_1), \ldots)$	Polygon
circle	$< (x, y), r >$	A circle (with center and radius)

Figure 13: Example methodology for SDT representation (Rigaux et al., 2003)

2.11 Accurate representation of S

GIS data consists of two (2) main types of data: the Spatial data and Attributes data. The spatial data/information are normally used to provide the visual illustration of a geographic space. They are saved as either **raster and vector types**. Therefore, this data is a blend of location and value data, which can be used for example for rendering a map. Geographical objects are grouped into two main classes; ***attribute data*** and ***spatial data***. Attribute data specifies characteristics at a certain location (e.g. where, when, how much, etc.) whilst spatial data defines the physical locality of the object of interest. Nevertheless, representing these data in a ***format, the computer understands, entails classifying the data objects into layers*** based on the individual components with related features (the layers might be elevation, waterlines, topography, temperature etc.). However, for each layer, the data properties (example **projection, scale, resolution, and accuracy)** has to be set, by selecting suitable properties for the individual layers. Generally, two (2) distinctive data forms are put into consideration, when demonstrating spatial data in a digital concept; ***(i) raster data format – field based (ii) vector data format – object based.*** In **Gregory et al., (2009)**, raster data structure could be likened to laying a regular grid over certain areas of interest and expressing the geographical characteristics or features existing in each grid cell mathematically. The field (or raster) based approach, handles spatial information (temperature, altitude, rainfall etc.) and as gathering of spatial functions, that transforms space subdivision into an attribute domain. ***Raster is connected largely with image processing, remote sensing and dynamic modelling, and they are easily operated using map/spatial algebra (e.g. computing the product of geographically equivalent cell values within two or more datasets***). Most S when being processed are represented in the form of raster or vector data (also see section 1.1). The raster DT is a primary data type, which consists of a combination of cells and their value (depicting a coordinate and the values), sometimes linked to an attribute table. For instance, the clustering techniques in SDM according to Jingbiao and Shaohong (2010) accepts a sample matrix, taken as input value, which we might think of simply as a point in the characteristic variable space. ***The information space is treated by the object-based method as if it is filled with distinctive, spatially relevant and identifiable objects.*** A representation of a S model from the perspective of Object-Relational (OR) databases comprises a set of SDTs and the procedures performed on these types of data. Vector (***object-based***) data structure, denotes geographic entities with the basic rudiments (***lines, areas*** (polygons---- (Figure 26 (b); Figure 10) and ***points)***. Vector data (as in **Gregory et al., 2009),** is constructed by considering point locations (of dimensions zero (0)) using their ***coordinates (x, y…), saved in two columns (for the 2-dimensions - x and y) of a classical/traditional database.*** A relational database (created by assigning each feature or item on

the table, a unique ID) can be employed, to connect location to an attribute table, elucidating what is found there. All the elements in a vector model are mathematically elucidated based on points that are well-defined by Cartesian coordinates according to **(Neuman et al., 2010**). S can also be denoted as incessant surfaces (e.g. pollution, precipitation, temperature, elevation, noise and so on), by utilizing the grid or raster data Model, where a mesh of square (or rectangle) cells is set over the landscape (that is the underlying object) and the value of the variable, described for each cell. In our study, we applied both data structures in the modelling of spatial databases because we were looking at creating generic spatial database algorithms. The S content we have explained in section 1, coupled with the proposition in Densham and Goodchild (1989) portrays spatial objects' exact geographic orientation and spatial distribution in the real world. S includes additionally, location, space entities attributes, the figure and their common relations. The recorded data can be the value of road length, height, point, building volume, polygon area (Figure 26 (b); Figure 10)), and the pixel gray. It can even be the string of a geographical name and annotation. Similarly, this data could be images, graphics, multimedia, spatial relationships or other topologies. Spatial phenomenon is normally designated using dimensional objects such as lines, points, polygons (area---- (Figure 26 (b); Figure 10)), thus the complexity of S and its inherent spatial relationships limits the usefulness of orthodox DM methodologies for hauling out spatial patterns. Figure 11 is a classic instance of a spatial dataset. The figure is an Image (Satellite) of a Region (the wards in Yorkshire and the Humber for the 2011 census) displaying the Region's borders (dashed white line). The Census-block with population, area, name, Water bodies (dark polygons) and Boundary (shown by dark line) as acquired from Census (2011).

For an *n-dimensional underlying space* R^n with a Euclidean distance, let us suppose that n =2, likewise, suppose the space of interest is a big polygon with its edges corresponding to the axes of the coordinate system (as in Figure 10 (a)), we acquire our spatial dataset (that is, all the elements or unique objects within the enclosed polygon). Hence, to save the data in a spatial database table, that is, construct a spatial database; we create a table i.e. relation, with the item sets acquired from the image using a relational database framework. If we haul out a chunk of data (i.e. a single block Census area table in its 2-dimensional space (see Figure 12)), we might reveal the information on individual pieces of records. Figure 14 is a classic example of a single object, in a suitable form to be categorised as an item on the database table. Traditional (classical) database according to Shekhar and Chawla (2003) does not have provision for storing boundary DTs (Table 2). Thus, there is the necessity to build a new relation (separate table) using spatial data framework (boundary column on Table 2) and map the new table to the classical (relational) database, hence, requiring the creation of additional tables (see Figure

12), that can save the SDTs. For individual study regions (rectangular block in the study area, Figure 10 and Figure 11) the following are identified; *polygons (*Figure 26 (b); Figure 10), *edges and points, and a* discrete relation (table) is constructed for them as in Figure 12.

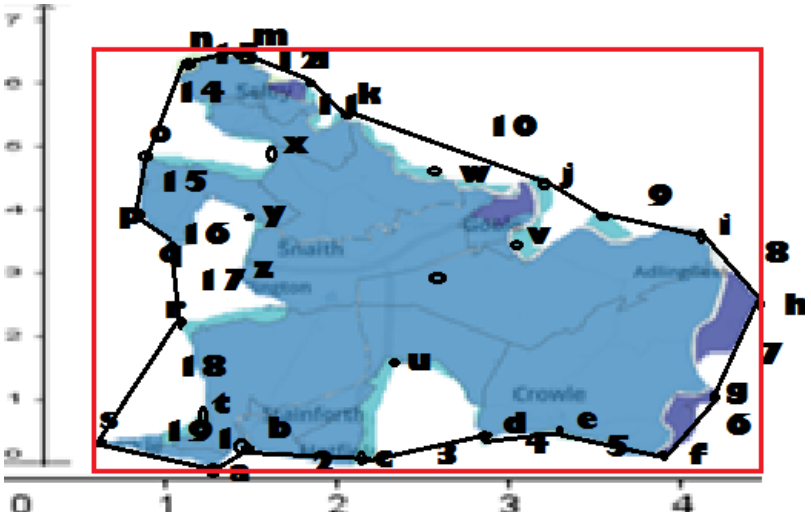


Figure 14: Typical spatial data representing a single object in its MBR (digitized from a bigger object consisting of the towns)

2.12 System Design for SDM purpose

A SDBS is a DS that deals with spatial and/or temporal DTs in their query language and data model. They normally have provisions for SDTs in their framework, yielding at the minimum spatial indexing and effectual methodology for spatial join. The major steps in building and managing the database include:

- **Modelling**
- **System query**
- **Algorithms and Data Structures (tools for Implementation)**
- **Systems Architecture**

The optimum aim of SDM is to incorporate and additional broaden techniques of traditional/classical DM in several areas of study, so that the databases will be adequate for investigation and management of large and complex S. In this study, spatial **DMGT** supports us in discovering the relationship across spatial and non-spatial data, also to **construct** and **query a spatial knowledgebase. SDB** according to **Gueting (1994)** are DBS for the organization of S. SDMS algorithms are useful and important for finding hidden rules, regularities or trends in large SDs, e.g. environmental studies, traffic control, geo-marketing, (**Koperski et al., 1996).**

2.13 Basic Operations for KDD in SDBS

In **Ester et al., (1999),** a SD is identified as a relational database with an additional concept of spatial position and spatial appendage. The obvious expansion of objects and their location depicts an implied relation (table) of spatial neighbourhood. The effectiveness of several knowledge KDD algorithms for SDBS rely greatly on effective processing of their spatial neighbourhood relationships, because the neighbours of several attributes/elements must be examined in a single run of a KDD algorithm (**Ester et al., 1997)**. Therefore, the author presented an innovative approach to KDD in SDBs, in which the sole objective is to extend SDBS through the use of data structures and operations for effective processing of implied associations of spatial neighbourhoods. This approach, according to them, allows a constricted combination of spatial KDD algorithms with the DBMSs of a SDBS, hastening up both the design and the execution of spatial KDD models. They also went ahead to describe a set of elementary operations for KDD in SDBS, used for expressing several significant algorithms such that most of the significant queries in a relational database are demonstrated using the five (5) elementary operations of relational algebra.

CHAPTER 3. MODELLING and METHODOLOGY:

3.1 Our proposed system

This chapter is concerned with the description of the framework that underpins our research objectives and the methodology. We briefly explain the framework we adopted for controlling, planning, structuring and the process of our research programme. This framework (our **system development methodology**), is one of the methods of developing an information system in software design and technology. It is a specialised technique employed in searching for scientific truth, creating correct elucidations of real-world phenomena, and building efficient systems. Our work presents an innovative method in which variables from a given dataset are extracted in the form of spatial attributes/variables, if the object can be interpreted or represented as an *n-dimensional Euclidean* space (R^n). The extracted attribute data is then stored in an *n-dimensional* array and manipulated as spatial data with the n-dimension space. This paradigm helped us to analyse real world dataset in the form of spatially distributed cases in specific areas. Thus, we came up with some advanced tools/methods and algorithms for mining large and spatial/non-spatial datasets.

3.2 System Development Methodology:

The best possible route to achieving our research aims, based on the standard System Development Life Cycle (SDLC) is the iterative methodology. The methodology also known as the prototyping model is our choice approach to completing this project successfully. We have adopted this methodology primarily because it well suits our objectives. In Figure 15, we have given a detailed description of how this approach works in helping us achieve our aim.

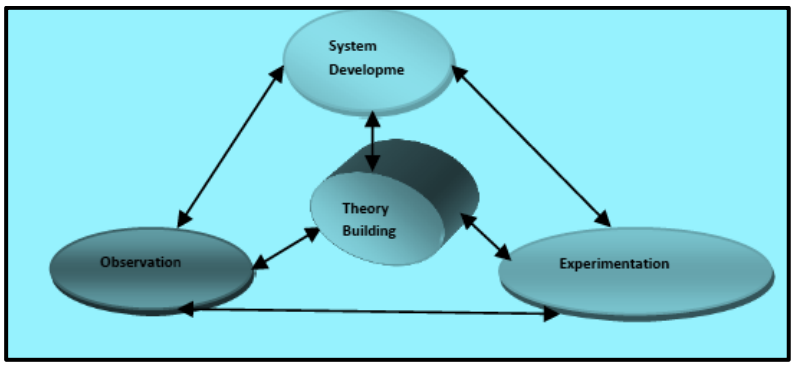


Figure 15: Various phases of the prototype model

The prototype software development methodology is popular and widely preferred for building a working baseline model. It involves sequences of bi-directional actions/stages that establishes the main frame of the technique. These actions/stages include: i) **Building** models with mathematical models, ii) **Developing** the system by describing all required elements. Note that this phase involves developing a new algorithm that would be compatible with the spatial database. iii) **Working out some experimentations**; for instance, via estimation or by exhausting field data and iv) **Observing** the performance and efficiency of the algorithm that might include using field studies, survey studies or case studies. We can further break down this approach as seen in Figure 16.

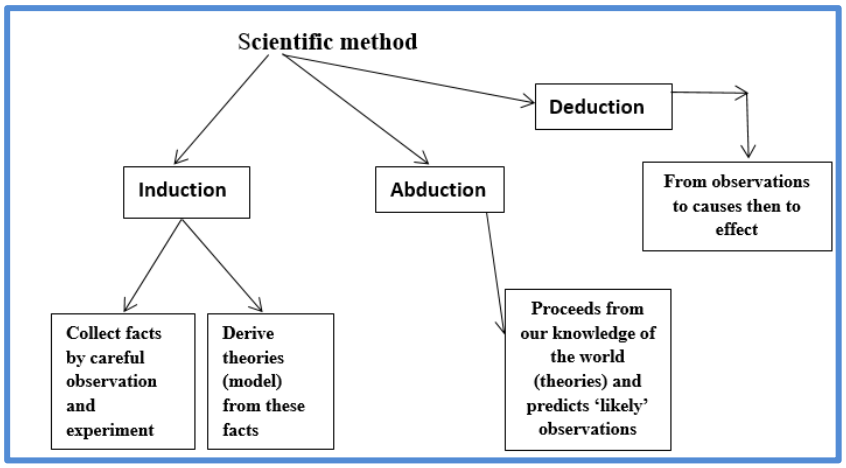


Figure 16: Expanded project research methodology as applied in this project

3.2.1 Application of Methodology:

In general, two main approaches were adopted to achieve our objectives (1) identifying the actual ***data types, data structures*** and ***spatial content*** of a given dataset (to make our model versatile and robust) and (2) ***integrating these data types*** into an appropriate **database management system (DBMS) framework, for easy manipulation.** These two approaches helped us to discover the general and varying types of patterns that exist within any given dataset non-spatial, spatial or even temporal (because spatial data are always influenced by temporal agents) datasets. We adopted **an iterative method** approach as it handles the non-linearity that always exists among spatial datasets.

We investigated new scientific and technological application domains where spatial DM methods and approaches could be used. We identified current challenges facing SDM. We identified current trends, procedures, techniques and application fields of SDM and we set out their strengths and limitations. We also carried out a detailed investigation on SDM in order to explore all of the

existing algorithms and their applications with the aim of detecting their strength and weakness. Our methods benefit from robust partitioning strategies, which minimises the excess and deep partitioning strategy common in most existing systems. The partitioning algorithms rely on a prefix-based approach for partitioning an activity set or search area into subgroups or sub-areas, so that individual sub-problems become solvable in the memory. We carried out experimentation and theoretical analysis on both simulated and real-world datasets and the result shows a massive improvement as regards older systems.

Following the approach elucidated above, we carried out our project by **a)** collecting necessary data (any type of dataset) which is carefully examined to express the data object as a feature space (as explained in section 3.1). If step one is successful**, b)** finding ***spatial patterns*** using maps or graphs. Finding spatial patterns simply means extracting possible spatial attributes from the objects variables as they relate to the feature space. Then based on the patterns discovered if any, **c) building a new model for each category** of spatial data mining tasks. finally, d) developing **the complete system**. After the implementation in d), a **series of experiments** is carried to test model performance using a series of test data (real world and simulated). If the result is satisfactory, then we **observe** the performance and efficiency of the algorithm that could involve using **field, survey, or case studies. In this study, two case studies were investigated, including i) SKIN DETECTION and ii) FACE DETECTION.** This study identified spatial factors in both quantitative and qualitative approaches. Our approach is not specific to the studied application case studies, therefore can be applied to explore the functioning of any ***technological, environmental, biological and scientific system affected by spatial and temporal processes***.

3.3 Spatial modelling (objects in space)

Geographical information systems (GIS) resources comprise of two (2) main types of data: Spatial data and Attributes data. Spatial data are normally used to provide the pictorial demonstration of a **geographic space.** They are saved as either **raster and vector types**. Therefore, this data is a blend of position and capacity data, which can be used for example for rendering a map. Location and extent (amount of ***space*** covered) are two (2) quantities that distinguishes spatial data from classical ones. The elements use spatial predicates (**inside, intersect, meets,** Overlaps, **adjacent,** contains, **encloses, near, far...),** to show the relationship that exists amongst the elements of any spatial event in a given environment **(*space*).** Although modern physicists usually consider space with an additional extent "time" which forms part of four (4)-dimensional continuum (space-time), physical space is often considered in three linear dimensions (Figure 18). Adam et al., (2008)

defines space as a three (3)-dimensional extent with no bound, where events and elements of an object occur, with relative site and direction. Thus, we state, everything around us including humans can be likened to an object in space. A space is a property described by the principle of Euclidean geometry (Solomentsev, 2011). In a more general term or logic, an Euclidean universe is a *fixed (n)-dimensional* **real vector** space R^n with an inner product **(e.g. y, z),** where **y, z** $\in R^n$, which in a suitably chosen coordinate (Cartesian – a rectilinear system of coordinates in a Euclidean space) system→ **y = (y1,…,yn) and z=(z1,…,zn) is given by the formula:**

$$(y, z) = \sum_{1=1}^{n} \quad y_i z_i \qquad \textbf{(1)}$$

Space can be considered in terms of **a) Geography,** Mazúr and Urbánek (1983), as one of the most fundamental of geographical concepts without an explicit and unambiguous definition, for which all geographical work must affirm. The concept of "geographical space" according to these authors is a relational one, conceived only as a supplement to things. **b) Mathematics,** as a term known "universe" or ("set"), with some added structure. In modern mathematics (Carlson, 2012, Aug), many types of spaces (Euclidean, Linear, Hilbert, Topological or Probability) are used, however, the notion of "space" is not defined. **c) Feature,** as the *n-dimensions* (as normally used in machine learning -ML) where your variables live. Hence, all variables in objects are usually viewed as features.

Our work depends on the third proposition, of space definition. Hence, given a problem of any nature (in form of an object), we assume the underlying space under consideration as a feature space of all the elements (columns, attributes, fields..) of the object. With this development, we extract all these elements into a table or *n-dimensional* array, where *the number of elements present determines n*.

3.3.1 Example of creating a feature space from an object

Given an object, say **PLANTS**

Follow the steps below to create a possible spatial model for the objects

1. **Identify the dependent (predicted or target) variable**

   Let the variable = Y= *species of predicted plant*

   Then:

2. **Independent (Predictor) Variables (elements of columns) could be:**
   - *X1= Temperature*

- *X2= Elevation*
- *X3= Precipitation*

3. **Following the logic in section** 3.3.1 step 2, the feature space is R^3. Thus, for the relationship, we could write a general equation of the form:

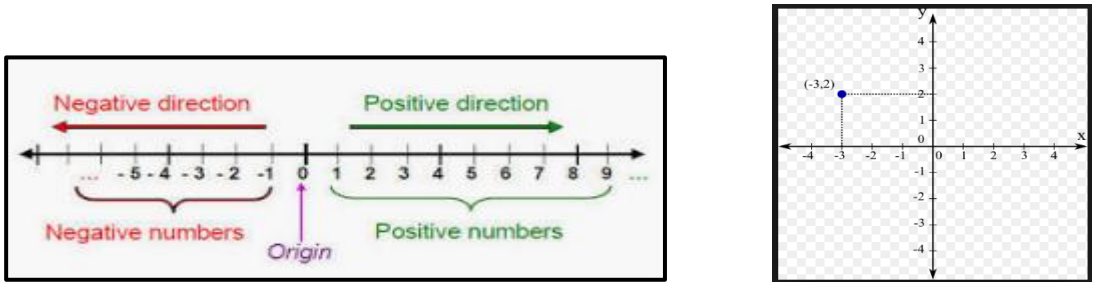
$$y\ (species) = \ \beta x \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (A)$$

→ $$y(species) = \beta x1 + \beta x2 + \beta x3 \ldots\ldots\ldots\ldots\ldots \quad \textbf{(B)}$$

Finally, we see feature space merely as the collections of features that are used for characterizing data. For example, in the plant data above, the feature space would be is the extent of all *n*-tuples of actual values representing the predictor variables i.e., **(βx_1, βx_2 …., βx_n).** This could be mapped to another feature space with a different set of characteristics describing the data, such as **(βx_1,** $2 * \beta x_2$ **…., βx_n-10**). **(C)**

3.3.2 Euclidean Space

In mathematical and algebraic analysis, Euclidean spaces (**ES**) can be defined from vector spaces, allowing the use of Cartesian coordinates (see Figure 17) and the power of calculus and algebra. Therefore, points are stated as tuples of real numbers (e.g. point one (1) = (value of x, value of y) named coordinate vectors. Similarly, geometric shapes are quantified using equations and inequalities that relate to or connect to the point coordinates. This method possesses the benefit of permitting the generalization of geometry easily to **ES** of three (3) or more dimensions. **ES**s can be modelled using Cartesian coordinates, as a choice of representation (i.e., real coordinate space (R^n) of the same dimension). In a single dimension, it is modelled as a real line (Figure 17 (a)); in two (2) dimensions (Figure 17 (b)), it is known as a Cartesian plane, with a coordinate space of three (Figure 18) or more real number coordinates, in higher dimensions, it is denoted by the *n*-dimensional Euclidean space R^n (Euclidean structure), with a finite dimension (Solomentsev, 2011).



(a) (b)

Figure 17: (a) single dimension, it is a real line$R$ (b) dimensional space (Cartesian plane---R^2)

According to O'Neill (2006), Euclidean (*n-space*), Cartesian space or just *n-space*, is the universe of all *n*-tuples of actual numbers, (w_1, w_2 …., w_n). Such an n-tuple is occasionally called points. The entirety of *n-space* is usually denoted as R^n. **Real coordinate space (RCS) (**in mathematics), of *n* dimensions, written as R^n, is a coordinate space allowing a number of (*n*) actual variables (elements or attributes) handled as a single variable. With different number of dimensions, R^n is normally used in many areas of physics as well as pure and applied mathematics. **RCS** is the ideal real vector space, frequently used for the Euclidean *n*-space **representation**. Thus, geometric descriptions R^n that are widely used for expressing the concept of space include plane for R^2 and three-dimensional space for R^3.

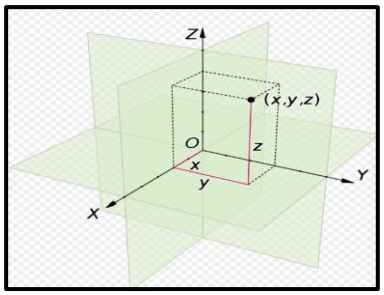


Figure 18: Three linear dimensions of physical space (R^3)

In geometry from the mathematical view of space, **Euclidean space** is said to encompass the two-dimensional Euclidean plane n=2 or R^2, the three-dimensional space n=3 or R^3or of Euclidean geometry, and related spaces of higher dimension *n* or R^n. The term "**Euclidean"** differentiates these aforementioned spaces (extent) from other types of spaces in contemporary mathematics. In a more general term, following the definition in Adam et al. (2008), the notion of "space" can be considered as regards the individual background elements, which forms an "environment" or "objects" filled with "qualities" which are the "attributes". Likewise, **Samson et al., (2014)**, defines spatial data analyses as the numerical study of phenomena that are situated in space. Based

on this conception, our spatial modelling applies the notion of space as a container or framework for positioning objects based on their position and extent and manipulating such objects based on their relationship with their neighbours.

Note: In space, Cartesian coordinates help to indicate an objects' position in space.

3.3.3 Procedures

Now we shall outline a practical example of the method described in section 3.3, we show how spatial data modelling is carried out. In **Procedure 1** below, we have modelled a procedure for a given system. Since the iterative design methodology, (adopted in this research work) is bidirectional, then with **the model building being** in the core of the process, we have flexibility of approaching the system design through any of the life cycle phases.

Procedure 1:

DATABASE = *PRESENTS EXISTING SYSTEM AND IT PROBLEMS*

DATA ANALYST = *EXAMINES DATABASE*

DATA ANALYST APPROACH PROBLEM = *BASED ON THE ITERATIVE METHOD*

APPROACH TO PROBLEM = *IS BASED ON THE NATURE OF PROBLEM*

NATURE OF PROBLEM = *MAY FALL INTO ANY OF THE THREE (3) MAIN CATEGORIES OF MINING TASK*

MINING TASKS = *MAY BE ANY:*

- *CLASSIFICATION (**Location Prediction - Where will a phenomenon occur**)*
- *ASSOCIATION (**Spatial Interaction - Where sub clusters/groups of spatial phenomena interrelate**)*
- *CLUSTERING (**Hot spots - Where locations or groups are uncommon**)*

Procedure 1.1:

IF CLASSIFICATION PROBLEM = DO

{

1. **Identify** attributes from objects
   - Test for the impacts of ***cause and effect***
   - Use appropriate data mining/statistical tool (graphs, maps, or any other visualization tool) to determine existing pattern
   - Determine events/dependent variables (DV) and event predictors/independent variables (IV)
   - Identify event predictor with strongest ***cause***
   - Define a suitable function to model the relationship between ***cause and effect*** variables
2. **Dispersed** spatial features from non-spatial features ***(Spatial features characterizes position, distances and neighbourhood,*** - They also express the spatial **position** and **range of spatial** objects***)***
3. **Consider** some of the spatial predicates that relate the spatial object to other spatial or non-spatial objects ***e.g. cross, close to connects, intersects, equal, intersect, disjoint, overlap, touch, within, inside, contains etc.***
4. ***Explore data using maps, other visualization***
5. **Define** a **learning set** (space) using all the attribute **(IV)** and then build a **training set** by selecting samples from the attributes

   E.g. given a spatial dataset $\mathbf{Y = \{y1, y2 ..., yn\}}$, let $\mathbf{y_i \in Y}$, where $\boldsymbol{i = 1 \text{ to } n}$, denote the attributes of **Y** or geographical location of the elements (point or a shape/extended object) of **Y**. A function $\mathbf{g(y_i)}$ may denote the attribute value of $\mathbf{y_i}$ **and h** $\mathbf{(y_i)}$ may denote the neighbours of $\mathbf{y_i}$**.** The values of $\mathbf{y_i}$ depicts the variables/tuple expressed in section 3.3.2.
6. **Build** a model or a novel spatial algorithm to manipulate the ***DBMS*** for future prediction of next location of occurrence.

   Note: the algorithm should be able to access any spatial database using an appropriate query language
7. **Verify the** result by testing **the model**, **Refine** and then **visualize the output pattern**

   Also note: the existence of spatial auto correlation would be a constraint

}

Procedure 1.2:

IF ASSOCIATION PROBLEM = DO

Similar to the definition in Agrawal et al. (1993), in association or spatial interaction problem, we want to determine consistencies between the elements in large transactional databases by finding from *transaction* data, *every rule that satisfies the least support* and the *minimum least constraints*.

Therefore:

1. **Given** a spatial dataset **Y = (y1, y2 ..., yn) as in *Procedure 1.1***
2. **Let a** set of transactions **T = (t1, t2, .., tm)**

Note each T_i is a subset of T

Then

- **Define an** association **rule** (for all Y_i) in the form $Y_i \rightarrow Y_j$ **for each** T_i;

Where $y_i \subseteq Y,\ y_j \subseteq Y,\ and\ y_i \cap y_j = \emptyset$ **(C)**

In that case events Y_i and Y_j are *autonomous* (with event Y_i having no influence on the probability of event Y_j), therefore, conditional probability of event Y_j if event Y_i *occurs*, is basically the probability of event Y_j, that is $P(Y_j)$

$$\rightarrow P(Y_j \mid Y_i) = P(Y_j) \qquad \textbf{(C1)}$$

Otherwise

The ***conditional probability*** **(P_c)** of Y_j is the probability that the event will occur if the knowledge that Y_i has already occurred. This probability is written $P(Y_j \mid Y_i)$, notation for the *probability of* Y_j *given* Y_i.

Note *for each sub-space of Y say* y_i, y_j, *by applying some spatial predicate* ***(inside, intersect, meets, Overlaps, adjacent, Contains, encloses, near, far...)****, this rule can become* $(Y_i => Y_j)$... **(D)**

This means that if the assumption in **(D)** *is true:*

Then

Given **(P_c)** *i.e.* Y_i ***given*** Y_j then Y_i and Y_j are not autonomous, and the probability of the intersection of Y_i and Y_j (the probability that both events occurred) is estimated by:

$$(P_c) = P(Yi \mid Yj) \rightarrow P(Yi\ U\ Yj) \rightarrow P(Yi\ and\ Yj)$$

$$\therefore P(Yi \mid Yj) = P(Yi)\ P(Yi \mid Yj)$$

$$\therefore P(Yi \mid Yj) = \frac{P(Yi\ and\ Yj)}{P(Yi)} \qquad \textbf{(E)}$$

Then

- **Assume that** there exists an association of the form:

$Y_i => Y_j$ *iff the computed value of* **(P_c)** in **(E)** = *minimum value of* **(P_c)**.

3.4 General procedure for modelling data as a spatial Dataset

Following the elucidation in section 3.3, in this section, we have outlined the basic steps for modelling a given dataset of any type and nature as a spatial dataset.

- Gather the data from a source (survey, observation or digitized map layer)
- Identify all attributes data (independent or predictor variables) present in all the available objects
- Test for auto-correlation between predictor and target variables (using Moran I for single independent variables and statistical packages for multivariate data)
- If autocorrelation is present, identify the type of pattern (clusters, variability, associations….) that exist
- Derive variables (using appropriate tool) that are most likely the strongest event predictors
- Test for the impacts of ***cause and effect***
- Build the prediction model: This aspect is achieved through explaining some event occurrences. By this, analysing and exploring the data done through

  a) Forming a set of hypotheses about these variables, which are likely to cause these events

  b) Testing statistical significance of the hypothesis and

  c) Modelling more precisely, any quantitative nature of the relationship that may exist using **linear regression** or any other tool for multivariate data.
- Finally, evaluate your model

3.5 Summary of methods and modelling for SDM

Lastly, in this section, we shall summarize the procedure for modelling an object, using SDM techniques. We have looked into methods of interpreting a given problem or an object as a spatial dataset. We also looked at the concept of "space," and how it applies to SDM. We described a simple step-by-step procedure for SDM task, we talked about the dimension of an object and finally, we discussed a simple and efficient way of extracting features from an object and projecting each attribute or element of the object as an item in an item *n-dimensional* space.

CHAPTER 4. PROPOSED METHODS and MODELS: DESIGN, IMPLEMENTATION, EXPERIMENT AND EVALUATION

In this chapter, we suggest a new method that improves the efficacy of the current existing spatial models/algorithm. Our incentive here is to enhance the performance of these algorithms w.r.t their techniques for interpreting and analysing large databases.

4.1 XDBSCAN: A PROPOSED ALGORITHM FOR IMPROVED CLUSTERING

Our proposed models gear towards the enhancement of spatial indexing structures by applying an improvement to an existing structure (the X-tree) by means of building a sort-based algorithm that will accelerate the operations of the original X-tree. The proposed model aX-tree is described in section 4.2 and the proposed application model (XDBSCAN), is described fully in section 4.1.5

4.1.1. INTRODUCTION

It is established in Li et al., (2016) that a great number of large datasets are spatially referenced; as such SDM remains the best means of accessing these data. Spatial databases are designed to manage the numerous applications, which deals with the emergent large quantities of data that are acquired from satellite images, X-ray crystallography, medical images text documents or other automatic equipment (Ester et al., 1996). Spatial relationships are very complex (Li et al. 2016) therefore it is often very difficult to manage large multidimensional databases which in most cases perform sub optimally in terms of user query processing as such leaving users with limited access or capability to examine large datasets (spatial or not) in detail. In support of the above argument, (Tian et al., 2015; Secchi, et al., 2015; Eldawy et al., 2015) have claimed that the present challenges faced by SDM is because all those existing relational and SDs are not effectual. SDs are built to manage data that come from large geographical information system infrastructure, therefore to query these databases efficiently based on user defined parameters, it is important to utilize a suitable indexing technique that will deliver an effective access to high-dimensional data for structuring the databases and further improve information retrieval. In many cases, these multidimensional databases normally map their multidimensional objects to some attribute (feature) of vectors in high-dimensional space and then queries are executed on a database of those attribute vectors (see explanation of feature vectors in chapter 3). According to Chakrabarti and Mehrotra (1999), similarity search based on extracted features or feature vectors, is emergent as a significant search paradigm in (database system system) DSs and the approach employed according to them is to map the data elements into points in a high-dimensional feature vector space that is then stored using a multidimensional data structure. In addition, domain-specific

entities (objects) found in many applications according to Patrick et al., (2015) are easily contrasted/compared and classified when they are represented as high-dimensional feature vectors. Similarly, to identify object sameness and to count obvious groups, most analysts frequently picture the vectors clearly in order to recognise clusters. Similarly, based on the elucidation in Samet (2006), Searching in high-dimensional spaces is time-consuming when it comes to performing similarity queries. Because the common goal of clustering algorithms basically aims at the evaluation of the underlying measures of similarity between data objects (Assent, 2012), data analysts have come up with several clustering algorithms to handle the problem associated with information retrieval in high dimensional spatial databases. One of these algorithms is the DBSCAN clustering algorithm. DBSCAN is an effectual clustering algorithm for SDSs, which detects outliers and noise, cluster randomly shaped point dataset and does not need to know the number of clusters in advance. Nevertheless, the DBSCAN deteriorates in performance when the size of data grows so large, additionally, the algorithm does not perform optimally if the incorrect values are selected for *min-points* and *eps* (the two significant parameters of the algorithm, explained in section 4.1.4). We suggest new methods and techniques that can improve the effectiveness of the existing DBSCAN clustering algorithm. The incentive here is to enhance the performance of the algorithm in terms of evaluating large (big in terms of number of attributes and number of cases/instances) spatial databases and in its method of selecting the right *min-points* and *eps* values. These limitations of the DBSCAN, as we mentioned earlier, could be averted if the existing algorithm is adjusted. Therefore, we enhance the implementation of the existing DBSCAN clustering algorithm for high-dimensional datasets to facilitate information retrieval processes in large spatial database by using a modified X-tree (**which we called aX-tree)** indexing structure for indexing multidimensional spatial dataset in a high dimension space, before performing the DBSCAN clustering task.

4.1.2. MOTIVATION:

It has been established that Index structures used in clustering algorithms perform excellently when the dimensions (d) of dataset (dimension of the data set is the number of attributes, especially the predictor attributes) is below 16 (i.e. d< 16), performance begin to degrades for d>20 to the level of sequential search (Berkhin, 2006). That means for more than 16 attributes, we consider the dataset as high dimensional. The major motivation behind this project is geared towards speed and precision in information retrieval from large multidimensional databases by the use of high dimensional spatial objects feature vectors. The R-tree based index structure (does not need point transformation for storing S) according to Berchtold et al., (2001) demonstrates efficiency, for the task clustering spatial datasets (a vital issue in measuring the performance of tree based algorithms

for indexing) however, they are not suitable for high-dimensional datasets. The index structure allows high overlap of the directory bounding boxes, which grows with increasing dimension. The complication here is that a good number of large SDs are often interpreted by the means of high dimensional feature vectors (see section 3.3 for feature vectors). Therefore, since feature spaces (section 3.3) always inclines to comprise several instances of related objects according to (Samet, 2006), then the database built with such a feature space is bound to be clustered. As such, indexing the database with an R-tree structure, might lead to cases of redundancy in searching the rectangles, because of the high overlap among MBRs of the R-tree nodes. According to Mamoulis (2012) numerous indexing structures (X-tree, A-tree and VA-tree) are proposed, which performs than the R-tree for indexing high dimensional data, nevertheless, a good number of them degraded in performance as dimension upsurges (Berkhin, 2006; Berchtold et al., 2001; Mamoulis, 2012). Therefore, going by the aforementioned premises, we present an improved DBSCAN algorithm, which is enhanced using an adjusted version of X-tree (**a-Xtree).**

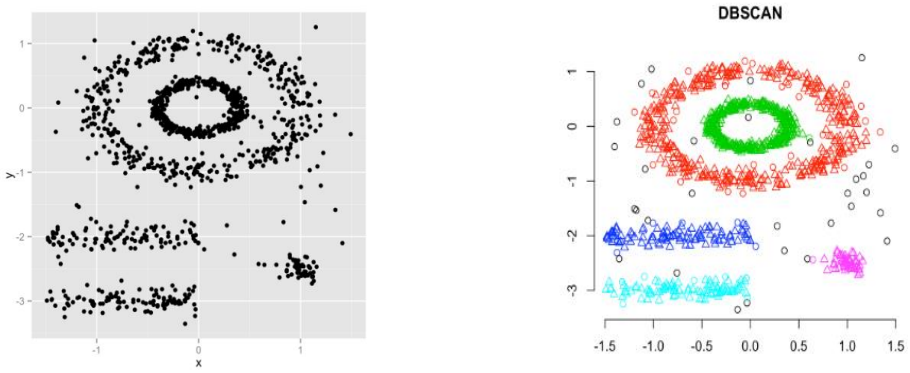
4.1.3. JUSTIFICATION:

According to Sander et al., (1998), spatial access methods are able to boost the performance of the existing DBSCAN clustering algorithm and make it more efficient even for large spatial databases. Though the DBSCAN algorithm has been implemented with so many other (tree) indexing structures, the X–tree has so far not been tested on this clustering algorithm. More so, because the X-tree is very efficient in handling high-dimensional multidimensional datasets, it might considerably increase the workings of the existing DBSCAN algorithm as regards its' time complexity and high dimensionality management (by combating curse of dimensionality – where there is no basically no clear distinction between similarities and dissimilarities). Also, the X-tree has been known to outperform other index structures for high- dimensional nearest neighbour queries owing to its 'super-node' features which avoids unnecessary splits and thus preserve appropriate neighbourhood locality, based on this, the operation on nearest neighbour queries particularly on high-dimensional (for example finding similarity between actual spatial objects) is enhanced. Therefore, we justify our choice of method as the best method for improving DBSCAN.

4.1.4. BACKGROUND AND RELATED WORK

A. Background:

Ester et al., (1996) defines the DBSCAN clustering algorithm as an extremely popular and widely used clustering algorithm because it is highly efficient in performance if applied on a SD, which would have otherwise resulted in unevenly shaped clusters. DBSCAN is density-based, therefore, performs well for finding homogeneity between highly clustered distributions of spatial objects. The algorithm equally performs adequately for clustering data without having a previous knowledge of the total actual clusters in the dataset. It also performs effectively in noise filtration from a dataset. The most significant aspect of the algorithm is that the generalised version, GDBSCAN (Sander et al., 1998), clusters spatially extended and point objects (based on their spatial and non-spatial attributes) efficiently. Notwithstanding, the DBSCAN has some key drawbacks, which include (1) High time utilization in discovering the neighbourhood of a data point/object (Szczuka et al., 2010). (2), performance degeneration with increase in dataset size (Vijayalakshmi and Punithavalli, 2012). The DBSCAN clustering algorithm groups a set of data elements/points based on the concentration of the elements in a given area. The logic behind the algorithm's density depends on two main parameters (***MinP and Eps***). The algorithm works by estimating the **Eps**-neighbourhood of each of the given points. The Eps-neighbourhood of any point, say ***p,*** is the set of points positioned around the Eps-distance of that point**.** ***p*** is distinguished as a **core point** if at least ***MinP*** points are found around its' **Eps-neighbourhood**. Other points are categorised as non-core points. Likewise, all non-core points have two distinctive groupings **(border or noise point).** Unlike noise points, border points are a non-core point containing a minimum of one core point in its Eps-neighbourhood. For the DBSCAN algorithm, the group of reachable core and border points from a particular core point creates clusters. When the algorithm discovers an unvisited core point, it analyses it and initiates a cluster by validating and stabilising the initiated clusters as a new cluster based on the value of its Eps-neighbourhood. The clusters can be extended by finding the Eps-neighbourhood of each point that is contained in the cluster; the process is continued until all points that are reachable from the first core point are found. That is to say, that the performance of the algorithm largely relies on the value of the chosen **Eps**-neighbourhoods (see figure 19).



(a) (b)

Figure 19: (a) Original points (b) DBSCAN cluster with **Eps** = 0.15 and ***MinP*** = **5.**

A. CLUSTERING IN SPATIAL DATABASES:

Clustering algorithms are very useful according to Ester et al., (1996), for the purpose of identifying object classes in spatial databases. Clustering in spatial data mining according Salman et al., (2013) means grouping closely related objects by concentrating on their relative density in space, connectivity, or the distance between them. In Bijuraj (2013), clustering entails the grouping of a collection of elements (objects) such that elements of the same group are more alike to each other than elements of other groups. In Assent (2012) Clustering has been defined as a mining task that is dedicated to the automatic grouping of data based on some common similarity. Kaufman and Rousseeuw (1990) have also defined clustering; Jain and Dubes (1988) as an expressive task that tends to detect similar groups of elements depending on the values of the underlying dimensions. In SDs, clustering (a) uses the measurement of the distance to an object nearest neighbour to identify the nature of the spatial relationships between the objects in those populations. (b) Searches for similarity among objects in a spatial database. (c) Algorithms are very useful for the purpose of identification of class in spatial databases (tends to detect similar groups of elements depending on the values of the underlying dimensions). (e) Groups similar objects together (see Figure 20).

Clustering algorithms for SDs according to Ester et al., (1996), specifically deals with job of class identification, this however does not come cheap as such an algorithm is expected to meet three (3) fundamental expectations

a) Having an elementary domain definite knowledge in order to define the input variables

b) Discovering of clusters with random shape from the large spatial database

c) Having a strong performance on large databases.

Though clustering has been applied to performing task such as gathering similar documents e.g. finding genes and proteins with related functionality or browsing, it has also been able to provide a grouping mechanism for spatial locations, which comes as a basic underlying framework for many dedicated application area (Steinbach et al., 2004).

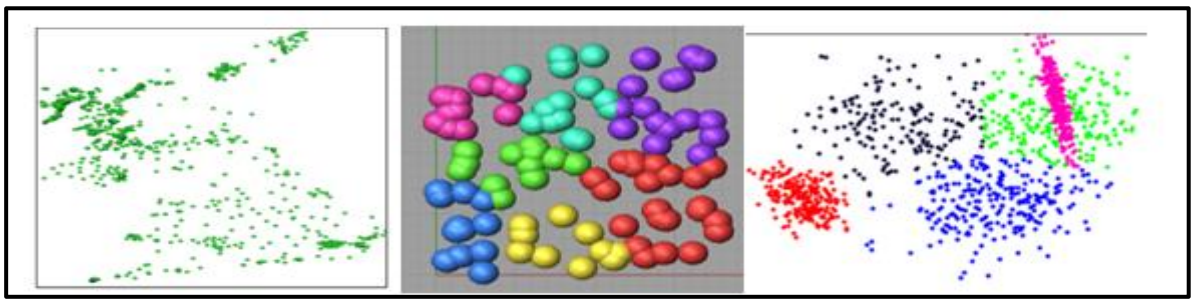


Figure 20: Sample clusters.

B. CLUSTERING AND HIGH DIMENSIONAL SPACES:

Some examples of application (more in Samson et al., (2013)) areas where the data (may not be spatial) is of considerably higher dimensionality incorporate, image databases, pattern recognition. Where the data consists of a set of elements, and the high dimensionality originates as a way of describing each of the elements using a group of features (feature vector – see the explanation of feature vector in section 3.3). In Samet, (2006). Examples of such vectors could be colour, textures, moments, shape, and descriptions. High-dimension expresses a scenario where the total amount of the unknown variables— explained in section 3.3, to be approximated, is one or more orders of magnitude (Bickel et al., 2001) greater than the total amount of samples in the dataset. High-dimensional data for large spatial databases is taken to be data defined by a big sum of attributes, if this becomes the case, according to Assent (2012). Therefore, as the dimensionality grows, there is an imminent idea that computational difficulty grows, for existing clustering algorithms. In Bouveyron et al., (2007), most scientific fields regard surveyed observations as high-dimensional, therefore, for high dimensional (vector or feature) space of such nature, clustering continue to proof difficult due to the reality that the high-dimensional data are always fixed in various low-dimensional subspaces, hidden in the original space. Objects, measurements and events are sometimes represented by a vector of their attributes (like points in a multidimensional space) according to Steinbach et al., (2004), where in most cases each dimension implies a unique attribute or variable that describe the objects, then an efficient clustering algorithm, is expected to find a strong categorization of the data into non-overlapping classes. Therefore, Bouveyron et al. (2007) has suggested that algorithms for clustering high dimensional data are expected to perform

an estimation of specific subspace (a set of the dataset contained in the lower dimension) and the inherent dimension of each group.) It has been identified that in high dimensional data According to Parsons (2004), many of the dimensions are often irrelevant and tends to hide clusters in a distance that exist between each object (and ends up perplexing the clustering algorithm). Therefore, in order to overcome this complication in very high dimensions, **feature selection** (finding a subclass of dimensions on which to carry out clustering, by eliminating redundant and irrelevant dimensions) and **subspace clustering** (localizing search in order to expose clusters, which are present in numerous overlapping sub-spaces) or even dimension reduction techniques are employed effectively for enhancing the quality of clusters (Parsons 2004). Berkhin (2006) added that the objects in DM always have numerous attributes and therefore presents difficulty for clustering algorithms, as such the data (feature) is basically initially transformed either by standardization (regulate the format between the variables /attributes /features), feature selection, or dimensionality reduction before the main operation is carried out (Parsons, 2004; Berkhin, 2006; Steinbach et al., 2004). Finally, the homogeneity amongst the variables (Jiawei 2001; Jain et al., 1999), is established, based on certain distance measures over the several dimensions in the dataset, based on the fact that objects are interpreted as a vector of measurements, or as a point in multidimensional space (Parsons, 2004).

C. CLUSTERING AND NN SEARCH:

Efficient processing of Nearest Neighbour queries according Roussopoulos et al., (1995) requires spatial data structures, which benefit from the proximity of the objects to focus the search on potential neighbours only. NN (homogeneity) retrieval or searching according to (Cazals et al., 2013), is a common computational issue with important applications in various fields of study. Many spatial clustering algorithms according to Berkhin (2006) rely on the indices of spatial datasets in order to enhance quick search of the NN. Finding an object's NN is a common operation in spatial databases, it involves having a spatial object, like a point or a route, and trying to find another object in the same set with the minimum distance (Saberi and Ghadiri, 2014). In Mamoulis (2012), NN was reported as finding the nearest object to a point ***P1*** in a spatial relation/table when given a clear reference object **O** (i.e., retrieve from a spatial relation/table **S**, the NN to a query object say **O**). For some database implementations with high dimensional dataset, NN or KNN queries are crucial (Berchtold et al., 1996). Hence, the major apprehension for NN search is CPU-time rate, which is constantly higher because the search is necessary to order all the internal nodes on their min-max distance. Therefore, if the data on the spatial table is not sorted (Mamoulis,

2012), there would be a necessity for the NN algorithm, to retrieve all objects in the table, in order to report the NN to a query object **O**.

D. ENHANCED DATA CLUSTERING ALGORITHMS WITH INDEX STRUCTURE:

Any n-dimensional data structure (R-trees, binary search trees, X-trees, B-trees etc.), can be utilised as a general approach for indexing the data in a spatial database (Cazals et al., 2013). In most situations, R-tree based structures are used (Song and Roussopoulos, 2001; Rigaux et al., 2003; Ester et al., 1999; Candan and Sapino, 2010; Cazals et al., 2013; Roussopoulos et al., 1995; Kuan and Lewis, 1997). The structures are built; using coordinates of the objects MBRs (which covers or contains the spatial objects) as input. Clustering algorithms for a SD can be boosted for fast NN search if they are sorted/indexed (especially with hierarchical structures). Indices act as good substitutions for reduced performance induced by dimensionality (Berkhin, 2006). According to Ester et al., (1999), spatial index structures (e.g. R-trees) are often employed in a SDBM to hasten the process of queries (for example region or NN queries). If a SDB is indexed with an R–tree, the tree nodes assists to hasten the search processes (Mamoulis 2012), below we have given a brief howbeit step-by-step description () of how the R-tree (and other tree structures) use its node properties to find the nearest neighbour to a point or object.

E. ALGORITHM 1 –Improved Nearest neighbour search algorithm for a tree indexed SDBMs:

- *Given a set of points* ***P***
- *Let* ***Pb*** *be total bounding boxes indexed with tree*
- *Let* ***P*** *be the query object* ***(orange point in Figure 21).***

  Then:
- *For all* ***tree*** *node* ***a, b, c, d, f…..,(letters in Figure 21.).***
- *Let* ***d*** *be distance between the mid-point MBR of* ***V*** *and* ***P*** *(the query object)*

→ ***d =Distance (P, β),*** *where* ***β = mid-point of V.***

$$\therefore \text{Nearest Neighbour } S \rightarrow \{S: d \leq \min(d_j)\}\ \forall j=1 \ldots\ldots\ldots\ldots Pb \qquad (F)$$

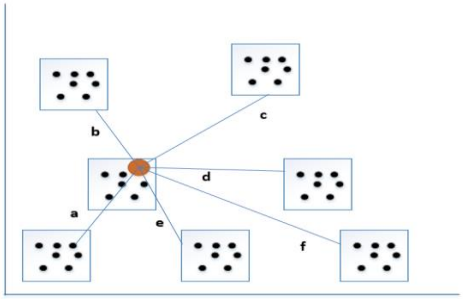


Figure 21: How to compute NN (orange coloured point is the query point)

E. ALGORITHM 1 –Improved Nearest neighbour search algorithm for a tree indexed SDBMs:

- *Given a set of points* ***P***
- *Let* ***Pb*** *be total bounding boxes indexed with tree*
- *Let* ***P*** *be the query object* ***(orange point in Figure 21).***

Then:

- *For all* ***tree*** *node* ***a, b, c, d, f…..(letters in Figure 21.).***
- *Let* ***d*** *be distance between the mid-point MBR of* ***V*** *and* ***P*** *(the query object)*

→ ***d =Distance (P, β),*** *where* ***β = mid-point of*** *V.*

$$\therefore \text{Nearest Neighbour } S \rightarrow \{S: d \leq \min(d_j)\} \; \forall j=1 \ldots\ldots\ldots\ldots Pb \qquad (F)$$

CHOICE OF K:

Several factors determine the value of **K,** including:

- Size of data
- Type and shape of data
- Task at hand etc.

In addition, the choice of **K,** defines the characteristics of a clustering or classification algorithm. Thus, choosing the value of **K** is one of the most core tasks of KNN algorithms. However, a more general approach for obtaining **K, is by experiment.** Experimentation is the most guaranteed method for determining the value of **K,** i.e., during implementation, run the module each time with a different value of **K;** record the optimal value for **K,** as obtained. By so doing, you can determine the best **K**, for value for your system

The description above resembles the branch and bound paradigm for NN search presented in Roussopoulos et al., (1995). However, contrary to their proposal, our method (the method we

described in section 4.1.5 of this chapter) only measures distance from ***O*** (query point), to say ***P1*** ***(***any other point***).*** ***T***he method in Roussopoulos et al., (1995), employs two techniques (mindist and minmaxdist) to order the NN search. The mindist measures the minimum (min) distance between the query point ***O*** and another point say ***P1.*** However, the minmaxdist measures the minimum (min) of the maximum (max) of the entire distances from ***O*** to any vertex (v) of the rectangle enclosing ***P1.*** Based on these measurements, the upper and lower bound of actual distance between ***O*** and ***P1*** is found and used for calculating the NN of the entire indexed elements in the data space. Kuan and Lewis (1997) has an improved version of the method for finding KNN that was proposed by Roussopoulos et al., (1995). The new method eluded the usage of the minmaxdist instead; they adopted the mindist only, as a way of eliminating redundant search for extremely correlated datasets. Their method (Kuan and Lewis, 1997) demonstrates an improved performance as regards the number of disk accesses for a single query operation however; the method is restricted by reduced computational strength. Samet (1995) suggested Quad-tree structures, for indexing SDs, because they easily overcome the complexity of location-based queries, thereby providing more effective answers to NN search by searching only the neighbourhood of the tree node that contains the query object (***O***). Nevertheless, research by (Nene and Nayar,1997; Candan and Sapino, 2010;) shows that in high dimension cases (data space, vector spaces and Euclidean spaces), NN search queries has failed to take advantage of available spatial index structures, due to cases of deterioration of search operation to linear scan of the complete database. Other tree structures (R*-tree, CR-tree etc.) according to Ioannidis et al., 2006 has also been used for indexing spatial databases for speed and better performance. Notwithstanding, the X-tree has essentially been designed specifically to tackle the problem with high-dimensionality, and has outperformed other spatial access methods. Berchtold et al. (2001) proposed the X-tree data structure. The structure has the properties below that assures a better performance compared to the rest:

1. The overlap minimal split nature of the structure and the presence of the super-nodes provide greater speed up for both point and NN queries.
2. The structure performs well for very large database sizes, with the increase in tree search time, which grows in a logarithmic O (log N- N is the total available objects) manner with the database size
3. According to Berchtold et al. (2001), the CPU-time of the X-tree is more than that of the TV-tree, R*-tree and others (because the NN queries need sorting on the min-max distance) however, it is still better than that of an R-Tree.

Like we mentioned before, objects, measurements and events are sometimes represented by a vector of their attributes (like points in a multidimensional space) according to Steinbach et al., (2004), where in most situations, each dimension indicates a unique attribute or variable that describe the objects, it therefore means that an efficient clustering algorithm should be able to find a strong grouping of the data into non-overlapping clusters. If any hierarchical data structures (R-tree, X-tree), is employed, to index the databases (Ioannidis et al., 2006), then, NN search algorithms may prune a branch of the tree, if it known that the area they cover, does not guarantee any element belonging to the NN of the query object **(O)** being processed.

B. Related Work

Enhancing the operations of the DBSCAN algorithm for huge datasets clustering has been the concern of many data analysts as such there has been a good amount of its variations. Szczuka et al (2010) stated, that one way to enhance the performance of the DBSCAN algorithm in terms of speed is through applying an indexing structure, which supports spatial data access method and therefore, hasten the neighbourhood finding operations (NN query) for the DBSCAN clustering algorithm. This concept has pulled a lot of attention and has led to disparity of the clustering algorithm; with a major aim to build a DBSCAN algorithm with different indexing structure that performs spatial operations efficiently. A distributed R*-tree (PDBSCAN) was applied by XU et al., (1999). The algorithm applied the tree structure for subdividing a dataset into various computer nodes. The Distributed R*-trees is useful for partitioning data however, the entire index is reproduced on each node. Amirbekyan and Estivill-Castro (2006) proposed implementation of the DBSCAN algorithm using the PP-Rtree. their version depends on the r-tree and as such, improves in performance if the database is first stored on an r-tree indexing structure. In Chen et al., (2010), another variation was proposed, the PDBSCAN. The PDBSCAN is a new parallel version of the original DBSCAN that works by executing a priority R-tree, this version is useful in a distributed environment. In (Vijayalakshmi and Punithavalli, 2012), the K-tree was employed to solve large database size indexing problems, particularly where the size gets too large (the main limitations of the existing DBSCAN). Their algorithm uses a k-distance graph method for automatic calculation of Eps and Minp values. Extended CUDA-DClust algorithm was applied in Welton et al., (2013). The algorithm (block tree index structure) for extending the operations of the DBSCAN. Mr. SCAN variation of DBSCAN algorithm is developed to handle serious problems in density based clustering, with a hybrid parallel tree-based implementation that combines an MRNet tree-based distribution network with a network of GPGPU-equipped terminals. Mr. SCAN partitions the data

space efficiently thereby, optimizing DBSCAN's computation over compact data regions, this practice helps avoid the problems met by earlier implementations. Hahsler et al., (2016) implemented the kd-tree, to aid the enhancement of the DBSCAN algorithm. The package DBSCAN is a fast re-implementation of numerous density-based algorithms built on DBSCAN technique, for S. Including LOF (local outlier factor) and OPTICS clustering algorithms. The implementation of the kd-tree index structure is meant to yield an accelerated KNN search. Chakrawarty and Gupta (2014) proposed the SR-tree based DBSCAN. This variation of the DBSCAN algorithm claims to work effectively in finding the Eps value of an object (the hardest task in running the DBSCAN algorithm). That means to compute the Eps value of a given spatial object (according to the author), its underlying region or space has to be computed, and then traversing the tree has to be from the children of the object to the leaves (Chakrawarty and Gupta, 2014).

4.1.5 CONSTRUCTING XDBSCAN ALGORITHM

A. Our contribution

For acceleration and enhanced performance, data mining algorithms largely depend on certain index structures. The choice of an appropriate index structure, likewise, depends on the type of query operation that the indexing algorithm needs to perform against a given database. Queries that need enhancement might include; NN queries, range queries, window queries etc. The R-tree and most of its variation have been widely applied to accelerate certain algorithms (including clustering) for various database query operations. However, research has shown that for dimensions greater than (>) 10, these index data structure degenerates to a linear search, which affects its performance. Thus, we propose a new way of improving and speed up the performance of the existing DBSCAN algorithm by implementing an adjusted X-tree index structure (**aX-tree)** in place of the R-tree. The X-tree was originally developed to build an appropriate structure for indexing/ordering spatial data and point data in high-dimensional spaces, consisting of multidimensional datasets. The eX-tended node (X-tree) proposed by Berchtold et al., (1996) demonstrates that structures like the R*-tree and a good number of its variations are not suitable for indexing high-dimensional data sets. Therefore, our proposed method proves to be an efficient method that accelerates the implementation of the existing DBSCAN clustering algorithm. It initially builds an adjusted X-tree (**aX-tree**) by following some procedures in Figure 22 and then implements the clustering algorithm by executing the aX-tree.

A. Method: The three (3) procedures proposed method

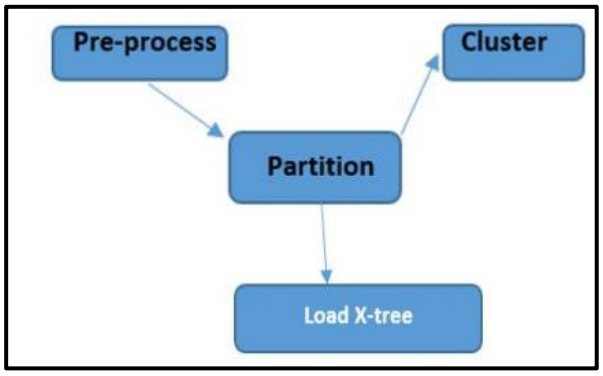


Figure 22: Project phases (based on the iterative methodology)

ALGORITHM 2a----procedure 1--------- data pre-processing:

1. *Given a set of spatial objects* Φ
2. *Let* Φ_j *be instances of* Φ, *where* $j = 1 \ldots\ldots \Phi$

   a. *Determine the maximum capacity of each node (say* g*)*

   b. *Find the midpoint of each object (for objects with extent)*

   Neglect step b, if objects are point data

   c. *Compute* $w = (\Phi / g)^{1/d}$, d*-dimension of the dataset*

   Where w *is the total item in the leave*

   d. *Then follow steps below:*

   1. *Sort* Φ *on the x-coordinates of* Φ_j *or x-coordinates of the centre of their bounding boxes (for extended objects)*
   2. *Partition* Φ *objects into* Φ_i *slices* $\longrightarrow$ *total of* Φ / g
   3. *Based on the ordering in 1*

   Then do ***PROCEDURE 2***

ALGORITHM 2b----procedure 2--------- Partitioning and Loading tree:

1. *Sort data in each* $\boldsymbol{\Phi_i}$ ***partition*** *based on their y coordinate*
2. *Load the* ***aX-tree*** *up starting from the leaves following the constraint below*
   -
   - *There is a maximum of* $\boldsymbol{g}$ *objects in the leave (*$\boldsymbol{g}$ *is computed in procedure 1) and*
   - *A total of* $\boldsymbol{w}$ *objects (from procedure 1) in a leave group.*
   - *Then load tree starting from the least* $\boldsymbol{\Phi_i}$ *object in the y coordinate and the* $\boldsymbol{\Phi_i}$ *object in the leftmost of the x-coordinate*
3. *Return bounding box of* $\boldsymbol{B[w]}$

ALGORITHM 2c----procedure 3--------- Data clustering:

A. *Determine a neighbourhood distance (***say** $\boldsymbol{p>0}$*)*
B. *Find neighbourhood value* $\boldsymbol{N>0}$*, from* ***PROCEDURE 2****, using the description in* ***ALGORITHM 1***
C. *Note:*
   - $\boldsymbol{N}$ *is a factor for determining the density of a cluster*
   - $\boldsymbol{N}$ *=(minimum)* $\boldsymbol{w+1}$*, to evade clusters having only one object*
   - *N is computed on each* $\boldsymbol{B[w]}$ *from* ***PROCEDURE 2***

Then:

i. *Find Bi=* $\{\boldsymbol{\Phi_i}$ *(from* ***PROCEDURE 1****)* $\in \boldsymbol{\Phi}: d(\boldsymbol{\Phi_i}, \boldsymbol{\Phi_{i+1}}) \leq \boldsymbol{p}\}$
d=distance between $\boldsymbol{\Phi_i}$ *and* $\boldsymbol{\Phi_{i+1}}$ *for i=1..... Φ*

ii. *If* $|\ \boldsymbol{Bi}\ | \leq \boldsymbol{N}$*,*
Then
Reject ***Bi*** *(an outlier)*

iii. *Compute the union* $\boldsymbol{Bi\ U\ Bj \longrightarrow IF\ \ Bi \cap Bj \neq 0}$*, for* $j= i+1 \ldots\ldots \boldsymbol{\Phi}$

iv. *Repeat* ***iii****, until there is no more union*

v. *Return: a set* $\boldsymbol{C}$ *of clusters*

4.1.6 RESULT AND DISCUSSION:

The proposed method in (section 4.1.5) holds a better performance than most existing clustering methods. The above argument is true because the X-tree hitherto has shown more efficiency in handling multidimensional data in high dimensional (feature) spaces (according to the review in this study). Additionally, the possession of the super-node is an added advantage, in the sense that the NN query will produce a quicker result devoid of overlap (**only at the leave, because there is a possibility of overlap at the internal nodes). In addition,** there would not be any need to search several rectangles (**within the internal nodes)** which obviously do not contain potential neighbours. Recall that though the R-tree based index shows high efficacy for spatial clustering, they are not suitable for high-dimensional space (see section 4.1.3) because the index structures allows high overlap (which increases with growing dimension) of the bounding boxes in the internal/**directory** nodes for a high dimensional data space. As such, due to the presence of multiple instances of similar objects, the database is certain to be clustered. Therefore, if an R-tree is used to index the database, there would be more instances of unnecessary searching of rectangles, as a result of the high overlap between MBRs (of the internal node) of the R-tree nodes. Using the new algorithm, we propose (Samson and Lu 2018), the overlap minimal split typical features of the a***X-tree*** will help to overcome the high overlap problem. From the algorithm (aX-tree), intuitively, the choice of ***minpoints and eps is*** easily determined following the description below:

Similar to the work in Roussopoulos et al., (1995), the calculation for these distances and values (***K, minp, eps***) is very simple. It is such that the underlying index structure supports tree pruning for any path that is not desirable in the search. Therefore, if the query point (**O)**, is positioned inside of the MBR (**see figure 23b**) the min distance (***mindst***) is zero (0), otherwise the ***mindst*** equals the square distance (Euclidean) between the point and the closest edge of the MBR containing the point ***P1***.

i.e.

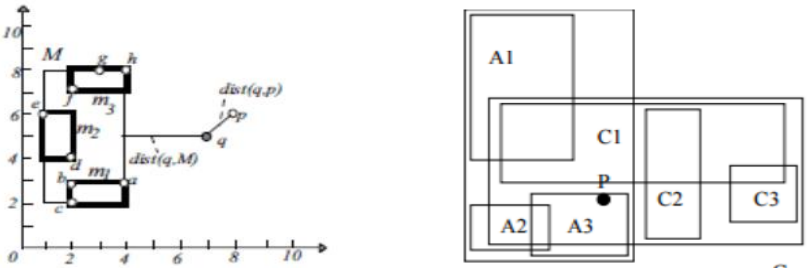
$$mindst = \{0, \quad if\ point\ (O)\ inside\ MBR \left(d(O,e)\right)^2, \quad otherwise$$

$$(where\ d = distance\ \ and\ e = the\ closest\ edge\ of\ MBR\ containing\ P1) \ldots \quad (I)$$

Statements:

- Derive the he eps value from the value of the min to max K-NN (deducted from the function of distance calculation as shown in Figure 23)

  *Note, we assume the **eps** value to be the value of the minimum **K** and the eps-neighbourhood **N** (--see* ALGORITHM 2c, section 4.1.5*), to be the values between the least and the largest **K** value in the NN search of **aX-tree***



(a) (b)

Figure 23: Computing NN using aX-tree nodes (MBR). (a) Distance between q (query point) and M- the MBR (b) Distance between p and the MBRs A and C, (query point is within the MBR, thus, least distance to p = zero) Kuan and Lewis (1997)'s theory.

- We choose the value of the k-distances (computed from the distances between the query point (**O**) and other density reachable points in the database) to represent the eps-neighbourhood (**N**) --*see* ALGORITHM 2c, section 4.1.5---,

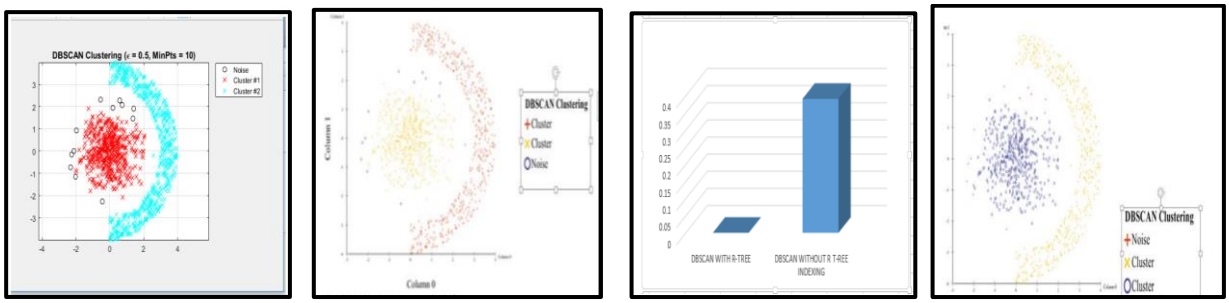
- We assume **k = max (k-distances) = kth neighbour = *minp.*** This means the neighbourhood contains k points plus the point itself which equal k+ 1 points.
  *Note we may not need to sort the value of the k-distances because the points were already sorted before loading the **aX-tree**,*
- We assume the result of the k-distances goes from smallest to largest (in ascending order to the largest k).

In Ester et al., (1996), it was claimed that ***minp*** can be eliminated by keeping it at 4 for all databases with data in 2 dimensions, however we disagree with this proposition, and suggest instead that ***minp*** should not be ignored, rather be modified because is the density determinant of any cluster. Therefore, we have assumed that k = ***minp. O***n the other hand, since we are adopting a bulk loaded X-tree, we can take the parameter (***minp) to be equal to either***

1. *at least **w + 1, w** is the total item in the leave (Algorithm 2a—section 4.1.5)*

2. *at least* the number of dimensions (in the dataset) plus 1 (as we have used in the algorithm), to eliminate any case of a single object cluster.

The intuition here is that any point less or equal to k-distances, will belong to the core point otherwise belong to the noise point or some other cluster. If we choose the ***minp*** to be equal to *dimension*+ 1, for say t dimensions, then we compute the KNN for the query point where max K = t + 1 (Then it means the total point in the neighbourhood N, of the query point equals t +2.



(a) (b) (c) (d)

Figure 24: Comparison between existing models (a) result of implementing the DBSCAN algorithm without an index, (b) DBSCAN with R-tree spatial Index, (c) DBSCAN with R-tree spatial Index plus sorting (d) DBSCAN comparison with and without the R-tree spatial Index.

Figure 24, shows a comparison of different methods for running the DBSCAN clustering algorithm. It shows that the accelerated algorithm, with improved performance, implemented with R-tree is faster than running the clustering algorithm without any indexing. As such guarantees that with the **aX-tree,** the **clustering** algorithm performs even better for higher-dimensional dataset

4.1.7 CONCLUSION AND FUTURE WORK:

DBASCAN is popular and well utilized in large databases for clustering big sets of data in order to discover significant and useful info from the data. However, research shows that the system does fail to operate effectively, when applied to databases with increasing size and volume. Additionally, the algorithm is said to have extremely poor worst-case complexity as regards discovering its two most significant factors, the ***Eps-value*** and the ***minp.*** In this section, we studied some existing methods/techniques for improving the efficiency of the algorithm as regards analysis of large databases and in terms of reducing its time consumption in the course of selecting the most appropriate ***minp*** and ***Eps-value*** values. We found that except for other spatial access methods that have been proposed for improving the DBSCAN algorithm, the R-tree index method and its variations have been the most widely used. Therefore, we propose a better indexing structure (**aX-**

tree) that has better potentials than the R-tree as regards processing data in high-dimensional space. The adjusted X-tree (**aX-tree**), similar to the original X-tree builds a spatial indexing method with minimum overlap splitting however; the new **aX-tree** in addition, is packed and pre-sorted, thereby retaining increased capabilities. The method reduces the time taken by the index to return the value of the NN query and as such hastens the clustering operation. For future work, we hope to carry out full implementation, testing and evaluation of the new algorithm.

In addition, better methods for computing the two most important parameters for the DBSCAN algorithm will be developed. Therefore, as our future work, we are working on creating new improved techniques for determining ***minp*** and ***Eps-value***.

4.2 AX-TREE: AN INDEX STRUCTURE FOR LARGE SPATIAL DATASETS

The major algorithm from this project is discussed in this section. We have presented a new technique (described in details in Samson et. al., 2018), for spatial indexing, which improves the existing X-tree S structure. The volume of data in spatial related databases are always enormous and are constantly growing larger. Therefore, minimizing the storage space utilization of these databases are of fundamental importance. The emergent nature of Spatial (location) data has led to growth in the size and dimensionality of SBs, also the highly pervasive qualities of these data make them to need precise, scalable, accurate, and cheap systems for high quality query processing. Numerous spatial indexing structures are proposed as regards achieving the needed quality. One of these structures is the *X-tree*. The *X-tree* is recognised as being highly capable of tackling high dimensionality in large databases. X-tree and most of its variations are built for a dynamic environment, with the ability to handle insertions and deletions. Notwithstanding, the tree structure portrays reduced performance on retrieval procedure as dimensionality grows thereby, leading to a bad worst-case performance than a linear scan *as the number of dimensions grows higher than 16*. Thus, we present a new X-tree packing technique (described in details in Samson et al., 2018) for static SDs that can perform optimally in space usage via careful packing. This new structure provides two (2) basic advantage:

1) Reduction of the X-tree index space overhead (by employing the functionalities of bulk loading --tree packing--. Tree packing entails computing the size of the node before loading data, to avoid wasted storage)

2) Yields a good return time (since the X-tree has higher fan-out and so the tree ends up shorter).

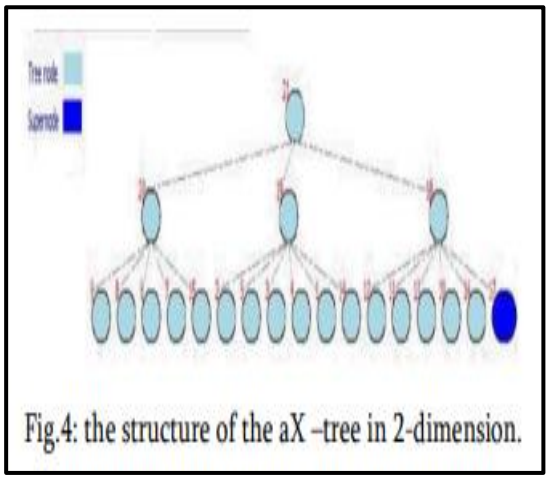
4.2.1 Introduction:

SDs are enhanced for querying and storing data (objects based on their geometric space). Tian et al., (2015); Mokbel et al., (2004); Vieira and Tsotras (2013), claimed that recent big data research has mainly paid attention to *spatial and temporal data.* The authors elucidated that this is simply because these kinds of data track *the position and behaviour of an event or object over time.* Nevertheless, the ever-growing accessibility to S and the daily eruption (in the volume of S) generated from numerous gadgets (space telescopes, medical devices, smart phones and many others), demands dedicated systems for handling big S. In spite of tremendous effort in SDM research methodology, several current studies have shown that there are still delicate aspects and matters of location/time DM that still demands attention. According to Kamlesh et al., (2015), Kang et al., (2014), Mauder et al., (2015), Billings (2013), Meng et al., (2014), Eldawy et al., (2015), Tian et al., (2015), Moussalli et al., (2014 and 2015), Secchi, et al., (2015), and Huang, and He (2015 and 2014), these problems include *traffic-aware navigation, location privacy, inconsistency, uncertainty and scalability in managing* S. According to the authors, all these spring from factors that are often causes noisy in the *location/time* data and inaccurate solutions to these problems are sought on daily basis. Query processing, indexing, visualization and language (main components of big spatial data namely) according to Eldawy and Mokbel (2015) are the main concerns to deal with in the study of mining large spatial datasets. The need advanced systems for handling spatial or spatiotemporal data is that, while traditional big data is highly aided by a selection of systems; cloud infrastructure (Hive, Hadoop, HBase, Spark, Dremel, Vertica, and Impala) and Map-Reduce-like systems, not any one of these systems delivers any special aid for spatiotemporal or S. Thus, to aid big spatial data, one of these measures must be taken

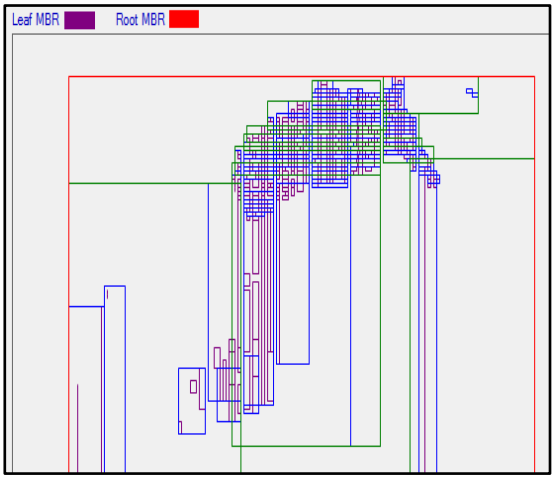
(1) Handle the dataset as non-spatial data or

(2) Write functions like wrappers around existing systems for non-spatial data.

Nevertheless, taking the aforementioned measures, results in a below average operation according to (Eldawy and Mokbel, 2015, June), because the systems do not take advantage of the properties of spatial and spatio-temporal data. Therefore, on the above synopsis, we present a new packing technique for X-tree **(aX-tree), suitable for** static SDs that works well in space management via careful packing. This proposed model has two (2) primary advantages: (1) reduction of space overhead of the index and (2) production of a good response time. These advantages are because **a**X-tree (Figure 25) has a higher spread (fan-out at the leaf node) and so will always end up shorter. Additionally, a new method for super-node creation and efficient technique for optimum packing based on an enhanced *str* bulk-loading method was also presented.



(a)



(b)

Figure 25: Performance measure of **aX-tree** (a) typical example of the tree in two dimension (b) result of packing 2D point locations of zip codes in USA

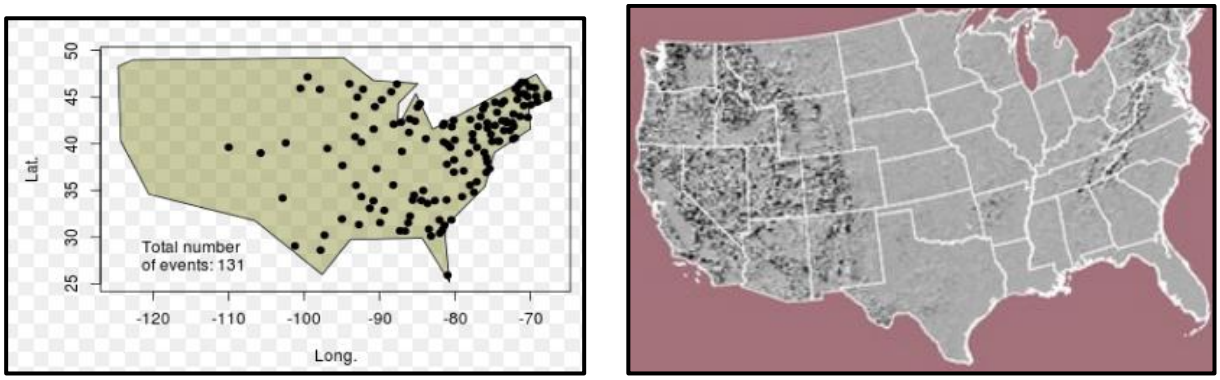
4.2.2 Background of study

A. Spatial data representation for study purpose:

Spatial data objects (Figure 10; Figure 26 (b)) according to Samson et al., (2018), mostly consist of multi-dimensional spaces, not well represented by point location (Figure 1). Therefore, index structures that support *n-dimensional* queries (range) depending on the object's spatial extent and location are constantly needed. Any problem associated with the attributes (spatial in this case) of a given object could be a typical query. Almost all spatial or non-spatial data features *(*Candan and Sapino, 2010), can be characterised as one or more form(s) of the four (4) standard basic models: *graphs/trees*, *vectors*, *logic-based*, *fuzzy/probabilistic* and *strings.* Nevertheless, due to the non - linearity common among large spatial datasets, an effective data structure with the ability to handle the diverged structures existing among spatial datasets is required. According to Candan and Sapino, (2010), these complex spatial dataset characteristics *are well characterised using trees and graphs. Similarly, because large datasets* often consist of other lesser events and objects that are hard to arrange in form of sequences. Such complex data includes

1. Hierarchical data (Xml-3-Dimensional worlds and taxonomies) that are always easily characterised as trees, and
2. Undirected or directed networks e.g. social networks (with the edges of the graph denoting obvious or implied connections between media objects or individuals).

Different variations of tree structures are utilised in specific applications for performance optimization. According to Patel and Garg (2012), evaluations between various tree structures are done with reference to, query kind support, *application, data kind support* and *complexity.*



(a) (b)

Figure 26: How to represent spatial data as points (a) location of part of a country represented as point, (b) actual spatial data polygon of the same map in a, with extent marked by city boundaries

B. Indexing with tree structures:

The aim of building index structures is to enhance the speed of query processing Ajit and Deepak (2011). According to Samet (2009), in large databases (particularly spatial temporal types), search effectiveness depends on the degree to which the given data is ordered (ordered). The ordering is handled by the indexing technique used for storing the S, therefore, making it more reachable. According to Cazals (2013), to store objects in these databases, every object is mapped to an attribute vector in a vector space. The attribute vector then functions as a portrayal of the objects. *The conventional role of these indexing structures is to order (sort) the data in the database. Nonetheless, there are no standard sorting techniques for data in dimensions higher than one, without one dimensional data transformation. As such, the sort algorithm, therefore, performs the function of distinguishing between the data. This differentiation/ordering simply entails arranging the spatial objects (in the table) as regards the occupied space*. The resulting ordering must be implied not obvious *so resorting might not be necessary, when the queries change (that is to say, the index need not be rebuilt for every new query). These aforementioned indexes order/arrange the space* (Samet 2009). Ordering tools like Hilbert curve, Z-order curve, or any other dimension or space reduction tool (that transforms a multidimensional data to single-dimensional), if applied to spatial indexing techniques (Park et al., 2013) transforms the index structure to an efficient optimization mechanism, for quality enhancement of large dynamic databases. One of the main characteristics of these sorting procedures (mentioned above) is the ability to project multidimensional datasets into one dimension, while conserving the neighbourhood of the data

point. After *the data is arranged/ordered (sorted), according to the ordering techniques mentioned above, a S structure can then be constructed on top of it, query results are cleansed, if essential, by applying the information from the initial attribute vectors*. For indexing, according to Cazal et al., 2013), any *n-dimensional* data structure is used *(B-trees, binary search trees, X-trees, R-trees* etc.). Nevertheless, extended objects (represented with rectangles---see Figure 10) are according to Guting (1994), more challenging to translate than points, because they often fail to fall within a single cell of a bucket partition, thus three (3) strategies are designed to handle rectangles in data partitioning for spatial index design including clipping, overlapping bucket regions and transformation approach.

C. Tree data structures for SDM:

In Table 3, different tree index structures are outlined. However, B-tree, X-tree and R-tree have proved to be the elementary and most used index structures. Notwithstanding, they have some drawbacks, which warranted their variations. High-level data models (particularly those involving *spatiotemporal* information representation, *links* or *object hierarchies) need graph-based or hierarchical representation*. Therefore, homogeneity-based classification and retrieval tasks normally would entail *comparing graphs and trees*. S objects (Samson et al., 2014), mostly consist of multidimensional spaces, not well represented by point locations (Figure 26 (a)). B-tree (a one-dimensional index structure) does not perform well with S because search spaces are always multidimensional. Therefore, an index structure that supports n-Dimensional queries (range) depending on the object's spatial extent and location are constantly needed. According to Patel and Garg (2012), evaluations between various tree structures are done with reference to, query kind support, *application, data kind support* and *complexity*. It was observed from their work, that for a B-tree of order *n (where n is the highest total number of children for each upper node*), the worst-case time complexity and space complexity of the tree is O (log *n*), for X-tree is O (*n*)-*particularly in some exceptional cases where the tree structure linearizes.* It is noted for R-tree that space efficiency is very poor. We can see that the ability of the X-tree to handle large high dimensional datasets is greater. These capabilities of the X-tree structure, can curtail most of the challenges currently faced by big "spatial datasets" management, however, not a lot has been said in latest researches, about the X-tree.

B-tree

The B-tree was presented in Bayer and McCreight (1972). The tree structure is meant for organizing and maintaining large sorted database. Many variations of the B-tree have hitherto

emerged in the literature for handling object-oriented data. In Hung-Yin (2008), B-tree was described as an efficient structure for processing point queries. However, it is not suitable for range queries and multi-dimensional datasets, therefore, one of the bases for the limitations of relational systems was their incapability to manage new emergent applications (e.g. CAD/CAM, Multimedia, scientific geographical and medical applications) with B-trees. In Patel and Garg (2012), B-tree was described as a structure having a variable amount of children nodes with predefined range.

On R-tree:

R-tree is a variation of the B-tree index structures, which was proposed by Guttman (1984). The structure was initially directed at manipulating geometrical data (line segments, points, volumes, and surfaces) in multi-dimensional spaces. *The major logic behind the R-tree data structure is the grouping of nearby objects based on an axis-aligned polygon covering their extent (their minimum bounding rectangle* --MBR). T*he "R" in R-tree means* ***rectangle***. According to Berchtold et al., (1996), the main issue with R-tree-based index structures is the high *overlap of the MBRs in the directory/internal nodes, which grows with increase in dimension. The R-tree* (Guttman, 1984) is an index structure for dynamic spatial searching. The tree portrays data objects in numerous dimensions (x,y,z…).

Table 3: Comparison between Index Structures

Index Structure	Query type	Data type	Complexity	Application
B-tree	Point query [1]	Linear data [1]	O(log n)	Apple's file system HFS+, Microsoft's NTFS and some Linux file systems, such as btrfs and Ext4.
B+-tree	Point query [3]	Linear data [3]	O(log n)	Most of the database management systems like IBM DB2, Microsoft My Sql, Oracle 8, Sybase ASE etc.
B*-tree	Point query [3]	Linear data [3]	O(log n) use space more efficiently than B+-tree	HFS and Reiser4 file systems
UB-tree	Point query, Range query [18]	Linear data, multidimensional data [18]	O(log n) but not feasible for multidimensional data	Multidimensional range search.
H-tree	Point query	Linear data	O(log n) utilize space more efficiently.	Ext3, ext4 Linux file systems.
ST^{2}B-tree	Range query, k-NN query [15]	Multidimensional data [15]	Work more efficiently for the moving object data.	Application with multidimensional data but now not use because other data structure outperform it.
Compact B-tree	Point query [4]	Linear data [4]	O(log n) but use space more efficiently than B-tree	In place of B-tree.

R-tree	Range query [1]	Multidimensional data [1]	Not utilize space more efficiently, not have worst case time complexity.	Real world application like navigation system etc.
R+-tree	Range query [16]	Multidimensional data [16]	Non overlapping data utilize space efficiently than R-tree	Multidimensional data object
R*-tree	Point query, Range query [9]	Spatial data, multidimensional data [9]	Implementation cost is more than other R-tree variants but robust in data distribution than other ugly structures.	Application with data in form of points and rectangles
X-tree	Range query [14]	Multidimensional data, High dimensional data [14]	In some extreme cases tree become linear and time complexity O(n)	High dimension data
M-tree	Range query, k-NN query [10]	Multidimensional data [10]	Not require periodic reorganization, time is less in construction.	k-NN query, application use multidimensional (spatial) access methods

Similar to the B-tree, the R-tree is height-balanced and dynamic too. Insertion and deletion operations can be carried out alongside searching on the tree. The data structures (R-trees) are built on top of B+- trees. *However, despite its wide acceptance, extensive use and popularity as a key indexing application for multidimensional datasets, its main limitations cannot be overlooked. R-tree splitting mechanism (achieved by utilizing only indigenous information) can lead to an ineffective directory structure with MBRs being prone to high rate of overlap, particularly for high dimensional space.*

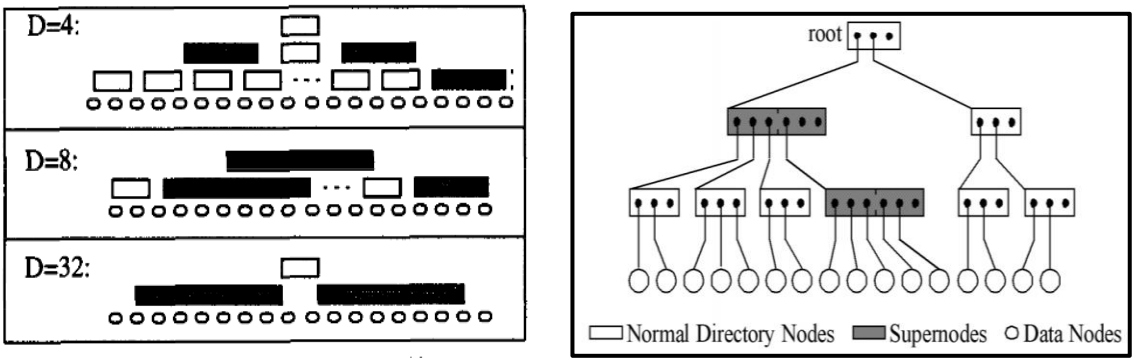
Limitations of the R-tree

The structure is highly suitable (Berchtold et al., 1996), for handling point and S, however, the performance of tree structure degrades fast with dimension growth, due to overlap in internal nodes, which rises quickly with dimension growth. The R-tree structure accommodates up to dimension five (5), with 90% overlap. Therefore, most recent research and improvements for the tree is grouped into two objectives:

- Bulk-loading (construct an efficient tree from the beginning or
- Improve the procedure for insertion and deletion, into and from the tree

In all, the aim of the final goal is enhancing the way the tree is constructed.

The X-tree:



(a) (b)

Figure 27: Original X-tree: (a) Shapes of the tree in different data dimension (b) actual structure of the X-tree in 2D, (Berchtold et al., 1996)

X-tree structure performs well for indexing point and S in high-dimensional space. The eX-tended node (X-tree– F**igure 27**), is proposed by Berchtold et al., (1996), as a method for indexing large volumes of point and S in high- dimensional space. Analysis according to Berchtold et al., (2001), shows that structures such R*- tree, R-tree are not suitable for indexing high-dimensional datasets. According to Ciaccia et al., (1997), X-tree and M-tree are practically variations of the R-tree. They are suitable for indexing multidimensional data. The implementation of the M-tree (based on some distance function (df) and triangle inequalities for efficient queries), is fully parametric according to the authors. The M-tree has no strategy to avoid overlap even though there is a high overlap of regions. Every node (n) of the tree is of radius (r), and each leaf node (l) that resides in n is at the highest, a distance (d1) from n. M-tree is a balanced structure that does not need intermittent restructuring. On the other hand, the X-tree avoids overlapping of inter nodes MBRs (the reverse is a big problem in high dimensionality). *If a node is not split, it may lead to a* super-node otherwise, for some critical cases the tree degenerates to a linear structure. X-tree (Berchtold et al., 1996), is a mixed structure of a hierarchical R-tree-like and a linear array-like directory. T*he upsurge in the spread (fan-out)* of the X-tree, according to Candan and Sapino, (2010), is the main positive side effect of the *super-node strategy.* The description of the X-tree given in Candan and Sapino (2010) Manolopoulos et al., (2010); shows that the tree is a heterogeneous structure because it consists of nodes of different types. During the processes of entering new data items into an X-tree, where it is impossible to avoid overlap, super-nodes are created. The super nodes explain for the advantage of X-trees over most other access techniques. Some *benefits* of the super-nodes include:

- Increment in storage utilization as a result of fewer splits taking place
- Reduced in tree height due to increment in average tree spread (fan-out)
- Sequential scanning of the datasets is for very high-dimensional spaces, where it is impossible to build a hierarchical access structure with minimized overlap between nodes bounding regions (MBR).

In addition to the "super-node", to *Stuller et al., (2000)* presented further description of the X-tree structure including the fact that the structure makes use of overlap-minimal, hierarchical internal model for low dimension, and a linear internal model for high dimensional vector. These structures steer to fast access of the object's attribute vector. Nevertheless, the tree degenerates and brings about extreme worst-case performance than sequential scan on retrieval performance as dimensionality increases, *when the number of dimensions grows more than 16*. In low dimension however, overlaps hardly exist between the triangles. *The X-tree employs the overlapping bucket regions that gain from having a key. Therefore, the key representing the spatial object might fall inside a single bucket; however, the problem here is that there are always numerous search paths, because of the overlapping of bucket regions.* Nevertheless, the tree structure is known to possess obvious drawbacks, including:

(i) Likelihood of an overflow of the 'super-nodes'

(ii) Redundant overhead caused by numerous disk access

(iii) For critical cases, the *tree* will totally become linear thus, leading to inefficiency in memory management.

This study pays huge attention to techniques and methods of constructing accessible large SDBS by expanding the structure of the existing *X-tree* model, thereby enabling it to overcome its limitations. Our proposition is a new heuristic based spatial indexing model; called the adjusted *X-tree (aX-TREE)* constructed on top of the existing model for effective handling of high-dimensionality in large data. Many variants of the *X-tree* (*VA-File, CB, FX +-tree, etc.)* are proposed however, none of these methods show complete efficiency for higher dimensional SDM.

```
int X_DirectoryNode::insert(DataObject obj, X_Node **new_node)
{
   SET_OF_MBR *s1, *s2;
   X_Node *follow, *new_son;
   int return_value;

   follow = choose_subtree(obj);       // choose a son node to insert obj into
   return_value = follow->insert(obj, &new_son); // insert obj into subtree
   update_mbr(follow->calc_mbr());     // update MBR of old son node

   if (return_value == SPLIT){
       add_mbr(new_son->calc_mbr());   // insert mbr of new son node into current node

       if (num_of_mbrs() > CAPACITY){  // overflow occurs
            if (split(mbrs, s1, s2) == TRUE){
                 // topological or overlap-minimal split was successfull
                 set_mbrs(s1);
                 *new_node = new X_DirectoryNode(s2);

                 return SPLIT;
            }
            else // there is no good split
            {
                 *new_node = new X_SuperNode();
                 (*new_node)->set_mbrs(mbrs);

                 return SUPERNODE;
            }
       }
   } else if (return_value == SUPERNODE){ // node 'follow' becomes a supernode
       remove_son(follow);
       insert_son(new_son);
   }

   return NO_SPLIT;
```

Figure 28: Algorithm for Insertion into the X-tree internal Nodes

```
bool X_DirectoryNode::split(SET_OF_MBR *in, SET_OF_MBR *out1, SET_OF_MBR *out2)
{
   SET_OF_MBR t1, t2;
   MBR r1, r2;

   // first try topological split, resulting in two sets of MBRs t1 and t2
   topological_split(in, t1, t2);
   r1 = t1->calc_mbr(); r2 = t2->calc_mbr();

   // test for overlap
   if (overlap(r1, r2) > MAX_OVERLAP)
   {
       // topological split fails -> try overlap minimal split
       overlap_minimal_split(in, t1, t2);

       // test for unbalanced nodes
       if (t1->num_of_mbrs() < MIN_FANOUT || t2->num_of_mbrs() < MIN_FANOUT)
            // overlap-minimal split also fails (-> caller has to create supernode)
            return FALSE;
   }

   *out1 = t1; *out2 = t2;
   return TRUE;
}
```

Figure 29: Algorithm for Splitting the X-tree nodes

The X+-tree:

The X+- tree permits increment of the size of super-nodes in the X-tree (Doja et al., 2012) to some degree. T*o avoid overlap (that brings about a bad performance), it is possible for the super-node increase while inserting data items into the tree*. Nevertheless, scanning large super nodes (linearly) is quite inefficient. For the X-tree, a super-node might be numerous times larger than a normal node in size. On the other hand, in the X+-tree, the case is different. The super-node size for X+-tree is at most the size of a normal node times a certain user variable (known as Max-

SNODE). According to Doja et al., (2012) if *the super-node grows larger than a certain upper limit, it is further split into more nodes*

The CBF and VA-File tree structures:

VA-File structure presented in Stuller et al., (2000), is a data structure introduced to improve the query performance and fight the degradation problem that is typical of the X-tree. These problems lead to a sequential scan of the X-tree nodes due to overfilled super-node in higher dimensional data. By applying a vector approximation transformation (Figure 30), the amount of disk I/O accesses can be decreased and the high dimensional vector space is subdivided into groups of cells that afterwards produces an estimation of individual cell, this venture enables the system to scan the VA-File for a candidate cell if a user sends a query. By taking this measure, the attribute vector within each candidate cell is explored to get its KNN

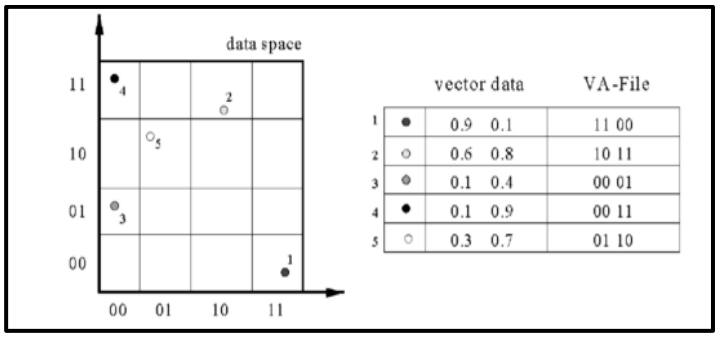


Figure 30: An example of vector approximation according to (Stuller et al., 2000)

4.2.3 Bulk Loading Tree Data Structure:

A. Sorting and Space Decomposition

According to Dolci et al., (2010), Partitioning is described as the decomposition of a search space in smaller relevant parts/units in order to achieve an efficient and fast information management (storage, querying and retrieval) system. The process is achieved by representing objects of the real-world type, as data items in the data structure of a spatial database or a geographical information system.

Data sorting

For the purposes of further processing, the importance of *location* according to Samet (2009), as a component of data, (that is a source of *enhancement for the value of the S* and *visualization)*, is never overstressed. The effectiveness of search algorithms largely depends on the degree to which the given data is ordered (sorted). Nevertheless, (Ajit and Deepak, 2011), *bigger datasets* emanate from lesser objects or events that are most likely difficult to sort into sequences. Thus, Samet

(2009) suggested that because no sorting exists usually in dimensions greater than one (1), without a one (1) dimensional transformation of the dataset, *then, the sort algorithm therefore performs the role of differentiating between the data. This differentiation/ordering simply entails arranging the spatial objects (in the table) as regards the space the objects occupied.* Sorting/ordering in this case, is a way of constructing an X-tree algorithm that takes the advantage of pre-processed data, before storage. Pre-processing is usually essential because

1) It yields an opportunity for good space utilization for known static datasets

2) Only a few nodes need to be accessed, while performing a query, thus, there is always a guarantee of improved query processing time.

3) Without pre-processing, algorithms (dynamic) that enter the data objects one after the other do not perform brilliantly for query processing (Leutenegger et al., 1997).

Sorting normally indicates the presence of an ordering. They are very suitable for one (1)-dimensional datasets. E.g. for individual data items, we could sort the items by their height, weight, salary etc... For two-dimensional datasets and greater, unfortunately, there are no such solutions. For example, if we order or arrange all of the towns in the USA according to their distances from Washington, the process will identify the nearest town to Washington efficiently; for instance, with a population larger than three hundred and forty-two thousand (342,000). Nevertheless, the same ordering cannot be applied to finding the nearest town to Los Angeles, with a population larger than one hundred and fifty-two thousand (152,000) for instance, without resorting/re-ordering the towns table. The difficulties experienced with two (2) and greater dimensions (Preparata and Shamos, 1985), is that the logic behind sorting is not in existence, except for dominance relations like Eq. A holds.

Given two (2) points g and ***h;***

Then

$(g) = \{gi \mid 1 \leq i \leq k\}$ *dominates* $(h) = \{hi \mid 1 \leq i \leq k\}$

$$If\ gi \leq hi,\ 1 \leq i \leq k, \qquad \text{Eq. A}$$

Where ***k*** stands for the dimension of the underlying space.

Nevertheless, according to Sagan (1994) and Samet (2006), with a space-filling curve, there could be an assurance of the presence of a sorting order by transforming the data to a linear structure. Regrettably, such explicit sorting order does not fit the necessities of having a dynamically sorted index structure that remains consistent even if the query changes.

The proposed structure (**aX-tree**), which is a modification of the X-Tree data structure, is expected to offer an enhanced partitioning paradigm for spatial objects based on two major heuristics

1. *Sort spatial objects based on their geometric qualities*
2. By using a *sort key*, divide sorted objects or MBRs into groups (buckets), which implies their physical storage region on computer) or into groups (Figure 31). For each of the **aX-tree** input elements (rectangle or point). The illustration in Figure 31 describes a typical scenario of the sorting and grouping (groups A and B) procedure. Given the points: 1…21**,** as shown in Figure 31, **C** depicts the grouping of the elements, based on the sorting strategy applied.

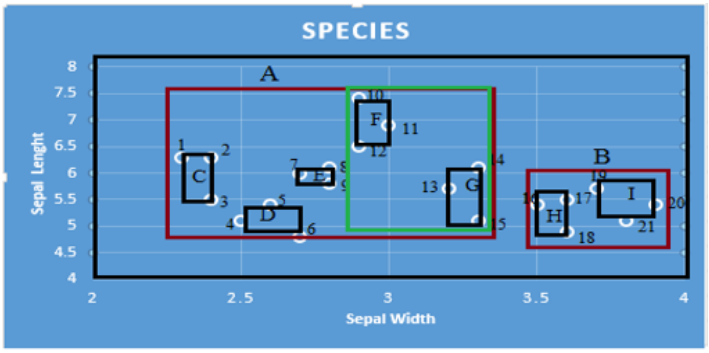


Figure 31: Typical input rectangles for spatial data analysis

Our basis for secondary sort lies in the fact that secondary sorting of individual groups never incurs additional I/O access (as long as the sorted groups, on the impending axis fits into main memory). Mamoulis, (2012) and Mamoulis, (2011) stated that bulk loading method for X-tree is expected to produce a minimal total cost Constraints. Our algorithm yields an enhanced query performance for all data types, ranging from mildly skewed point and region data, evenly distributed, highly skewed point and region data. *Additional characteristics of our proposed model include:*

- Maximal (or increased) node occupancy
- Lower height of the tree
- Hierarchical directory node quality
- Very reduced overlap within internal nodes
- Smaller MBRs sizes

Bulk-loading a Tree Data Structure

A bulk loading technique rather than the direct insertion operation typical of the existing **X-tree,** will help to overcome the sluggish approach of individual object insertion, recursively and possibly overcome the over expansion of the super-node. After the data is sorted, using the sort mechanism we have described above, a S structure is then constructed. Any n-dimensional data structure can

be used for indexing the data, such as B-trees, R-trees X-trees etc. (Cazal et al., 2013). Thus, we can overcome the difficulties of unanticipated overloading of the "super-node" that reduces the efficiency of the X-tree**.** Bulk-loading method constructs a tree at once, rather than iteratively inserting individual objects into an empty tree one after the other. An efficient bulk loading technique (Mamoulis, 2012 and Mamoulis, 2011), would construct rapidly for static objects and will guarantee a smaller number of wasted redundant spaces on the tree pages. According to Giao and Anh (2015), the advantages of bulk loading a tree structure is follows:

(a) Tree loads faster at once with spatial objects

(b) Empty spaces are reduced in tree internal nodes

(c) Improved splitting of spatial objects into tree nodes.

B. Partitioning:

According to Leutenegger et al., (1997), some algorithms already exist for packing tree structures (see Figure 32). These methods benefit the choice of partitioning strategy that we adopted for decomposing the underlying space, including:

a. **Nearest-X method**: in this method, proposed by (Roussopoulos and Leifker 1985), data objects are ordered by ("X"), the first coordinate of the data dimensions only and then divided into desired page size.

b. **Sort-Tile-Recursive (STR)**: this method was presented in (Leutenegger et al., 1997). The STR algorithm is a variation of the Nearest-X. It evaluates the total amount of leaves needed as in Eq. B

$$J = \lceil capacity\ of\ a\ node \rceil \quad \text{Eq. B}$$

Then apply the split factor on the individual dimension (d) using Eq. C

$$p1 = \lceil J^{1/d} \rceil \quad \text{Eq. C}$$

Then repeatedly divides each dimension consecutively into Eq. D

$$p1 = same\ size\ partitions \quad \text{Eq. D}$$

Using one dimensional ordering for each split.

If they cover more than one memory page, the resulting pages are bulk-loaded again using the same algorithm. The leaf nodes will not overlap for point data, and the data space would be "tiled/sliced" into roughly equivalent sized pages.

c. **The Hilbert Sort (HS) or Packed Hilbert R-tree: these sorting methods, described in** (Kamel and Faloutos, 1993), is equally another variant of Nearest-X, however, the algorithm sorts with the Hilbert value of the midpoint of the rectangle (MBR) rather than the X coordinate. There is no assurance that the pages will not overlap.

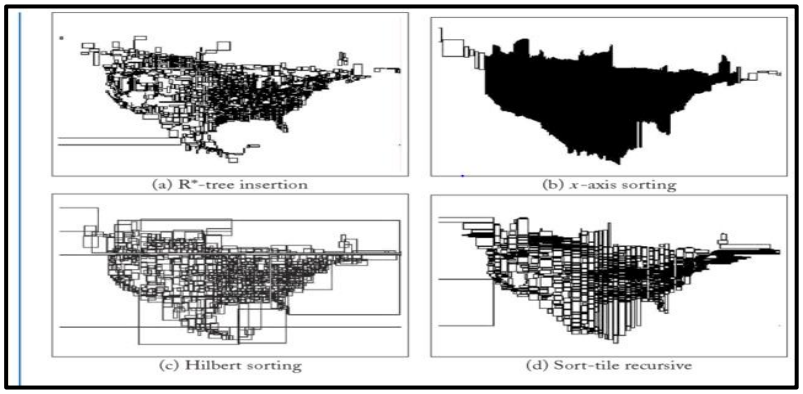


Figure 32: MBRs of leaf node of R-tree based on different sorting algorithm according to (Leutenegger et al., 1997)

C. Sort-tile recursive Bulk-Loading:

The STR method for secondary sorting, presented in Leutenegger et al., (1997), has the abilities of improving the efficiency of the X-tree, by eliminating the current straightforward indexing method. The algorithm leads to easy implementation of polynomial-time algorithms and might even exhibit greater efficiency with larger datasets as against the existing structures. Following the sort strategy above, the **a**X-tree is built by employing a bottom – up approach, using the sorted points (for point only data representation) or MBRs for extended data objects or bucket grouping, Below are some advantages of the STR ordering method

1. The method is an effective bulk-loading example method for the X-tree
2. It a normally applied according to Giao and Anh (2015), in GIS and DBMS environments
3. STR is yet to be used for enhancing the X-tree
4. Easy implementation, yet very good for query optimization

4.2.4 The proposed model (aX-tree)

I. Model description:

The algorithm we proposed is a new packing method (for X-tree), useful for static SDs. The work presented is different to the models we presented earlier. We give the assertions our proposed model is the first adjusted X-tree (for multi-dimensional datasets) that is based on secondary sorting methodology. Similar sort-based indexes (R-tree but not on the X-tree) models might exist for other spatiotemporal database management. However, other methods for handling dimensionality majors on *feature vector mapping, dimension or space reduction, point transformation, feature embedding, etc*. Similarly, previous methods are based mainly on enhancing the performance of the *R-tree,* however, only a small number of adjustments have been proposed for enhancing the *X-tree* (despite its abilities in managing high-dimensional spatiotemporal data Candan and Sapino, 2010; Manolopoulos et al., 2010; Patel and Garg, 2012; Jin et al., 2013; Dash et al, 2015). Our proposed system does not employ a similar insertion strategy as *X-tree.* We distinguish our technique by carrying out an initial sort (**STR**) before *loading (packing) the X-tree*. The proposed **structure** performs the filtering mechanism of *cost* reduction *(cost incurred from inspecting the geometric objects directly, necessitated by amplified overlap between objects' MBRs)*.

Therefore, a methodical arrangement (ordering) of the overlapping rectangles plus reduced extents of the MBRs may profit the query procedure as regards effectiveness and efficiency, since only a few MBRs are expected to intersect, is the main objectives of the proposed **aX-tree** model.

II. Motivation

It was observed in Shan and Wang (2010) that recent research on modelling focus mainly on modelling and sampling techniques themselves, neglecting the benefits of the features of the expensive functions that may be present. The aforementioned neglect often creates the issue of cost and computational complexity in high-dimensional data management. For example, it is known that for the existing X-tree, the most effective arrangement of the internal node is a hierarchical organization, in low dimensions, equating the height of the tree to the amount of essential disk page accesses. Contrarily, in very high dimensionality, a linear arrangement of the directory node becomes more efficient. Therefore, we assert that the X-tree is an ineffective structure for high dimensional data management. Because, while it is quicker to visit the linear part of the tree (avoiding multiple paths), the tree still bears *high implementation cost that results from the overhead produced by the super-node (particularly for cases where* the given query does

not cover the entire MBR of the super-node*). Similarly, the X-tree tolerates other costs, including the cost resulting from visiting the nodes of the hierarchical part of the tree. I*n some bad cases, the X-tree totally linearizes, thus, leading to an inefficient memory management. It is also known that in higher-dimensional dataset, many geometry-based data structures (X+-tree, X-tree etc.) fail to perform well. Existing variations of the X-tree are Cell/grid based indexing methods. These indexing methods require point transformations to store the S, thus are not good for spatial clustering. *These drawbacks of the X-tree motivate our new structure, adjusted X-tree (**a**X-tree).*

III. Packing the tree (aX-tree construction)

Packing should be overlap free between the spatial objects and their container rectangles. For some variations of packing algorithms, the main aim is to discover an optimum set up for packing a single container (MBR) with the maximum fill. In **Lodi et al., (2002**), the aim of these algorithms is to pack available data items into as fewer containers as possible. For some variations, the intersection of MBRs or containers of objects are permitted, however, must be minimal. We implemented an STR algorithm for an X-tree on a 2D plane.

This work is focused on (points). The method can also be extended to lines and regions by simply estimating their geometry, using one of two different approaches as described in Dolci et al., (2010) and Lee et al. (1996 September). (1) Using the objects' MBRs (smallest axis aligned rectangle enclosing the object), enclosed with the points {x_{min}, x_{max}, Y_{min}, Y_{max}}, so as to estimate their extent in space based on their maximum and minimum values for single measurements on individual axis (see figure 7). (2) Using an accurate object decomposition method, where the spatial object (complex) is broken down into simpler and smaller spatial components.

If approach one (1) is adopted, then the tree construction follows the steps described below. We represent each spatial object or query as a point ({***y-min, x-min, y-max, x-max***}) in 2D space so lines, points and regions can also be represented using their MBRs.

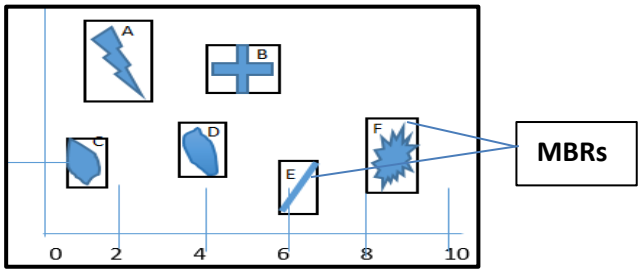


Figure 33: MBRs of spatial object

Step1: Pre-processing

First, the dataset is pre-processed, by sorting according to the illustration in section 4.2.3. For extended spatial objects, we consider the centroids (Table 5), of the spatial objects' rectangles (MBR---Figure 33), then move down to step 2 and then focus the partitioning on the two parameters x and y. In addition, to approximation and optimization techniques, we can also apply an initial randomizing of the entire dataset, which is conducted as regards large datasets before the sorting procedure (to reduce computational complexity). This additional technique is an improvement method (for pre-processes), to select only a significant sample of the original data before the partitioning. After a successful partitioning, the entire dataset can then be scanned and each object is placed in the right partition based on the chosen interval (range).

Step 2: Partitioning

Next, partitioning. Here the STR algorithm described in section 4.2.3 (c) is employed; however, in this case, the partitioning is done logically through an interval (*range)* partitioning procedure. Spatially close objects are packed together in one parent node. This assures that dead space in the parent MBR is minimal, and the parent MBR is compactly populated with child MBRs as possible. We assume the STR algorithm is applied to each dimension in a sort and partition manner
i.e., for a two-dimensional dataset, with x and y coordinate (Table...) where x, y depicts the midpoints for extended objects, or Table 5), where x represents sepal length and y represents sepal width.
The logical slicing of the space relies on the value of $\boldsymbol{R}$ in the formula:
$\boldsymbol{S = \sqrt{P}}$***; where,*** $\boldsymbol{P = \lceil R / c \rceil}$ and $\boldsymbol{R}$ is total point available and $\boldsymbol{c}$ is the node's maximum capacity.

The entire space is partitioned recursively until all the dimensions (x, y…n (dimension)) is considered.
The procedure is described below:

Step 3: Building the tree:

Given ***R*** data points (from Table 4 for point objects or Table 5 for extended objects)

Note: *Where data not point then*

1. *For all spatial objects find the 2-dimensional MBR around the object,*

Approximate the centre of each MBR (on the x-axis), based on the formula in **(**Eq. E

2. **)**
3. *Continue with steps below*
    a. Let ***R1*** be the size in kb of ***R***
    b. Let c =8kb, be the page maximum capacity for each node

$\therefore P = \lceil R1 / c \rceil$, is total pages to be consumed or number of leaves

$S = \sqrt{P}$, the total slices (on each dimension) of space depends on the value of S.

$$C_{x,y} = \left(\frac{x2-x1}{2}, \frac{y2-y1}{2}\right) \ldots\ldots\ldots\ldots \quad \text{Eq. E}$$

Detailed description of the stages

In this section, we shall be presenting a general summary and exhaustive description of the methods mentioned above

PROCEDURE 1: PARTITION

Given a collection of data (spatial object or point data) from a relation ***R*** (the tables illustrated above)

Let *C* = maximum capacity of a node //i.e. the size of available block storage in the computer

Let k = dimension

// the approximate total number of leaves estimated→ $P = R / C$,

The leaf nodes contain pointers to the objects in ***R****, they also contain pointers to the* ***MBR*** *for each object.*

Then follow the steps below:

1. *Initially, store the x, y values of the midpoints of the* **rectangle** *(e.g.* ***x-coordinate of the centre of extended object, if not extended objects, then store the points) in a table as in Table 5***

2. *Sort the values, based on one dimension (e.g.* ***x-coordinate of their centre for extended object) if not extended objects, then sort the points***

3. *Determine the maximum capacity of a node (**C**) stretching from 4kilobytes – 16 kilobytes page block size of the memory space (this will give you the total leave nodes -**i.e. S, number of pages at leave** level)*

4. *Organise the sorted MBRs into* $P = \lceil R / C \rceil$ *leave nodes (pages at leave level)*

//each group of P *- bounded by its MBR – will be arranged in the same leaf node.*

5. *Partition the sorted MBRs into* $S = \sqrt{P}$ *partition (vertical slices)*

// the vertical slices mentioned above is made up of $S * C$ *run of sequential rectangle obtained from the list of sorted rectangles*

//where dimensions > 2, then $S = P^{1/k}$ *for dimension =* k

//Therefore, we have $C * P^{k-1/k}$ ***run of sequential rectangles (slices or partition)***

6. *Sort/order the new **S** collections again, on the **y –coordinate** of the centre of the MBRs.*

7. *Repeat 1 to 5 until all the dimensions have been considered.*

Output 1:

*Following the packing of the **S** collections of MBRs into nodes, the output for individual leaf node (loaded into a temporary file for processing in phase two of the algorithm) is*

→ ***(MBR ptr, Node Id)***

Procedure two (2) works on the file (temporary) that ensued from procedure 1. In this phase, the **a**X-tree is recursively constructed going upwards (starting from the leaves nodes), until the root node is constructed.

PROCEDURE 2: BUL LOADING THE TREE:

Start:

1. Load the *groups of MBRs of sorted **S** groups from the file in procedure 1*

2. *Generate leaf nodes **(**base level **Lev = 0)***

DO

i. *Generate t**ree** nodes,*

ii. *Assign **C** MBRs to the node*

*// **C** is defined in procedure 1*

*//While node is constructed; evade any split that causes overlap, by expanding one **super-node** (only on leave nodes) in the current level, see Figure 34.*

***While** R /* in procedure 1*/ > 0*

Then

3. *Generate nodes* ***(Lev + 1)*** *at higher level*

While ***(nodes at level Lev > 1)***

a. *Order nodes at level (**Lev ≥0**) in ascending order of creation time*

Repeat ***steps 2***

b. *Return Root*

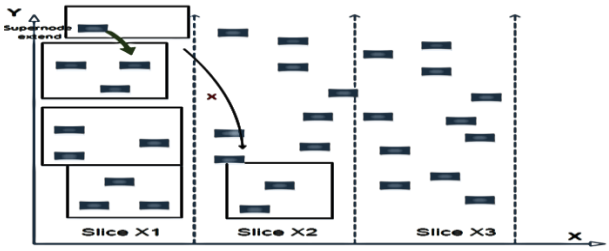


Figure 34: Showing the construction/extension of super-node

The Super-node:

Let us say we have a set of ***R***, *k-dimensional* rectangles (hyper), which may be overlapping in arbitrary ways. We would like to divide the area covered into groups of non-overlapping hyper-rectangles. We introduce the super-node. The super-node is a mechanism for shortening the height of the tree, with the benefit of also reducing the possibility of a linearized structure. R-trees make no guarantees about the efficacy of the bounding box (MBR). The only assurance is that each node in the tree will contain no more than ***S*** rectangles, where ***S*** is the block-size or node capacity (number of rectangles stored in each leaf on disk).

The simple heuristics in procedure 3, logically decides when it's appropriate to extend a node to supernode.

PROCEDURE 3: SUPER-NODE FORMATION

While creating the nodes in procedure 2

Do

1. *observe total number of objects (MBRs) in each partition*

2. *Determine* ***C*** *(maximum node entry for each node- see procedure 1)*

THEN

3. *For each partition*

a. *if remaining RECTANGLE <= **(C + ((C / 2) - 1))***

b. ***&&*** *RECTANGLE > **C***

c. ***&&*** *RECTANGLE**! = 0***

// RECTANGLE is used to represents objects that falls into individual partitions as in Figure 34

4. *Create **S** (maximum number of entries for super-*node)

{

***S** = **C** * 2;*

*→ **C** = **S**;*

}

END DO

4.2.5 Discussion and Experimental Analysis:

In this project**,** we carried out the actual experiment and evaluation of the core system (the **a**X-tree). The experiment reveals that the algorithm performs well for large and high dimensional static data provided as long as pre-processing is performed. In addition, points and extended objects are captured easily by simply constructing a convex axis-aligned MBR around the points, polygons or lines. The idea of building the rectangles around the objects (by constructing the smallest enclosing block around them based on the value of the geometry or geography column of the database table) is adopted to capture spatial objects in a simple but efficient manner without having to create separate methods for each different data type. For the experiment, several forms of data were indexed successfully, but only two different data formats and their outcome would be described in this section. The test data used for these experiments are real spatial point data. The first test consists of points in high-dimensional space (i.e. d = 2, 4, 6, 8, 12, 14….25), of the ***database with UK postal regions (see figure 35 (a)) with 1048575 rows and 169.016 MB data size*** database of uniformly distributed point data**.** Table 6 and Figure 35, show the performance of the tree structure with respect to increase in dimension.

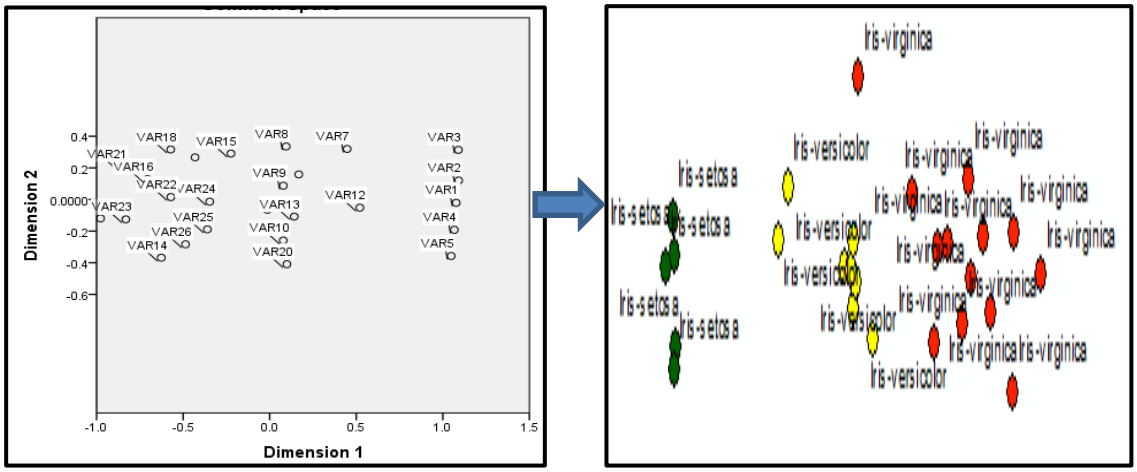
The second dataset used for the experiment is a database containing non-spatial data, that formed the database of the iris plant dataset from UCI machine-learning-databases website (UCI 1998). We used the multidimensional scaling (MDS) for pre-processing (see the stages in section 4.2.5 (A)) and final result in Table 4, to add spatial referencing to the dataset by casting the multidimensional data on a two-dimensional Cartesian coordinate planeR^2. We adopted this paradigm because we could represent the elements of the information in the database spatially, since there is a relationship between them. Table 5, depicts the relative position of the objects/classes (of the ***database with UK postal regions***) in a Cartesian coordinate system. The X

and Y values could be used to represent the same data in a GIS (equivalent to cities and other GIS point data). Using this method spatial coordinates (in this case X, Y) were extracted and assigned to the non-spatial response data. In the second phase we exported the data to ***Arcmap*** as tabular data and then the table was transformed to a point feature (***shapefile Table 5***). We performed further transformation by using the ***FME*** transformation tool to translate the ***shapefile (based on the extracted coordinate value)*** into ***ESRI*** standard data form and then constructed the ***geometry*** of the point feature. The geometry data was then exported to ***SQL Server*** as a geometric table of point data and then the indexing as described in ***Samson et al., 2018*** started. The MDS algorithm will place each of the data objects in a *k*-dimensional space in such a way that it tries as much as possible to preserve proximity between the object distances. Then each object is assigned coordinates in each of the *k*-dimensions (see result in Table 4). The number of dimensions of a plot (***MDS***) of dimension *k* can be above two. If we choose *k*=2 (like we have done in this project), then it means we are optimizing the object locations for a two-dimensional scatterplot. The same logic applies to three-dimension etc.

A. Procedure 1: Stages involved in obtaining a 2-dimensional MDS, from projecting the n-dimensional data unto a 2-dimensional plane, (b) actual locations of case on the 2-d plane

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	CLASS
	5.1	3.5	1.4	0.8	0.6	0.9	0.2	0.6	0.0	0.9	0.8	0.4	0.2	Iris-setosa
	4.9	3	1.4	0.1	1.0	0.5	0.4	0.5	0.7	0.3	0.6	0.8	0.2	Iris-setosa
	4.7	3.2	1.3	1.0	0.9	0.1	0.3	0.6	0.1	0.8	0.0	0.5	0.2	Iris-setosa
	4.6	3.1	1.5	0.7	0.1	1.0	0.7	0.7	0.9	0.4	0.1	0.2	0.2	Iris-setosa
	5	3.6	1.4	0.0	0.3	0.0	0.4	1.0	0.7	0.9	0.1	0.5	0.2	Iris-setosa
	5.8	2.6	4	0.4	0.9	0.8	0.6	0.1	0.5	0.2	0.7	0.8	1.2	Iris-versicolor
	5	2.3	3.3	0.3	0.7	0.6	0.9	0.7	0.6	0.7	0.3	0.3	1	Iris-versicolor
	5.6	2.7	4.2	0.2	0.8	0.6	0.4	0.1	0.4	0.5	0.1	1.0	1.3	Iris-versicolor
	5.7	3	4.2	0.7	0.4	0.5	0.4	0.4	0.0	0.8	0.2	0.1	1.2	Iris-versicolor
	5.7	2.9	4.2	0.4	0.4	0.1	0.9	0.6	0.4	0.5	0.6	0.3	1.3	Iris-versicolor
	6.2	2.9	4.3	1.0	0.1	0.4	0.0	0.0	0.6	0.6	0.7	1.0	1.3	Iris-versicolor
	5.1	2.5	3	0.0	0.1	0.6	0.9	0.2	0.1	0.1	0.7	0.4	1.1	Iris-versicolor
	5.7	2.8	4.1	0.7	0.0	0.6	0.1	1.0	0.6	0.4	0.4	1.0	1.3	Iris-versicolor
	6.3	3.3	6	0.1	0.5	0.1	0.9	0.5	0.5	0.9	0.6	1.0	2.5	Iris-virginica
	5.8	2.7	5.1	0.8	0.3	0.4	0.5	0.9	0.3	0.6	0.6	0.9	1.9	Iris-virginica
	7.1	3	5.9	0.1	0.8	0.7	0.9	0.5	0.3	0.9	1.0	0.2	2.1	Iris-virginica
	6.3	2.9	5.6	0.9	0.8	0.7	0.5	0.9	0.1	0.4	0.6	0.6	1.8	Iris-virginica
	6.5	3	5.8	0.4	0.5	0.1	0.9	0.1	0.2	0.6	0.5	0.9	2.2	Iris-virginica
	7.6	3	6.6	0.2	0.9	0.4	0.6	0.4	0.5	0.0	0.1	0.8	2.1	Iris-virginica
	4.9	2.5	4.5	0.7	0.5	1.0	0.4	1.0	0.2	0.6	0.2	0.4	1.7	Iris-virginica

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	CLASS	X	Y	Z
2	5.1	3.5	1.4	0.3	0.4	0.0	0.4	0.8	0.5	0.8	0.9	0.3	0.2	Iris-setosa	-1.125	-0.149	0.054
3	4.9	3	1.4	0.5	0.8	0.1	0.0	0.9	1.0	0.2	0.5	0.6	0.2	Iris-setosa	-1.135	0.023	0.03
4	4.7	3.2	1.3	0.9	0.6	0.1	0.6	0.5	0.5	0.7	0.0	0.6	0.2	Iris-setosa	-1.183	0	-0.108
5	4.6	3.1	1.5	0.9	0.2	0.6	0.6	0.4	0.5	0.2	0.8	0.5	0.2	Iris-setosa	-1.139	0.091	0.063
6	5	3.6	1.4	0.9	0.9	0.7	0.5	0.4	0.4	0.0	1.0	0.5	0.2	Iris-setosa	-1.134	-0.194	0.028
7	5.8	2.6	4	0.1	1.0	0.0	0.7	0.3	0.2	0.8	0.5	0.7	1.2	Iris-versicolor	-0.144	-0.004	-0.029
8	5	2.3	3.3	0.3	0.4	0.4	0.5	0.1	0.7	0.9	0.1	0.0	1	Iris-versicolor	-0.47	0.146	-0.056
9	5.6	2.7	4.2	0.7	0.3	0.2	0.4	0.6	1.0	1.0	0.5	0.0	1.3	Iris-versicolor	-0.093	0.048	-0.011
10	5.7	3	4.2	0.4	0.6	0.6	0.8	0.7	0.1	0.5	0.1	0.6	1.2	Iris-versicolor	-0.09	-0.077	-0.002
11	5.7	2.9	4.2	0.6	0.4	0.9	0.1	0.9	0.4	0.4	0.8	0.9	1.3	Iris-versicolor	-0.076	-0.029	-0.006
12	6.2	2.9	4.3	0.3	0.1	0.6	0.7	0.9	0.2	0.6	0.8	0.3	1.3	Iris-versicolor	0.024	-0.134	-0.006
13	5.1	2.5	3	0.5	0.7	0.3	0.0	0.3	0.2	0.3	0.6	0.8	1.1	Iris-versicolor	-0.529	0.049	-0.04
14	5.7	2.8	4.1	0.1	0.0	0.2	0.5	0.2	0.5	0.2	0.9	0.5	1.3	Iris-versicolor	-0.105	-0.009	-0.008
15	6.3	3.3	6	0.1	0.4	0.2	0.2	0.2	0.7	0.5	0.9	0.9	2.5	Iris-virginica	0.664	0.059	0.208
16	5.8	2.7	5.1	1.0	0.6	0.2	0.9	0.6	0.1	0.7	0.2	0.3	1.9	Iris-virginica	0.252	0.131	0.013
17	7.1	3	5.9	0.0	0.3	0.3	0.2	0.1	0.2	0.4	0.0	0.3	2.1	Iris-virginica	0.71	-0.085	-0.046
18	6.3	2.9	5.6	0.9	0.1	0.1	0.4	0.2	0.5	0.1	0.1	0.4	1.8	Iris-virginica	0.461	0.043	0.012
19	6.5	3	5.8	0.7	0.6	0.7	1.0	0.2	0.8	1.0	0.0	0.3	2.2	Iris-virginica	0.592	-0.024	0.026
20	7.6	3	6.6	0.8	0.3	0.6	0.4	0.7	0.7	0.3	0.9	0.6	2.1	Iris-virginica	1	-0.013	-0.057
21	4.9	2.5	4.5	0.1	0.9	0.5	0.3	1.0	0.3	0.2	0.7	0.5	1.7	Iris-virginica	-0.062	0.354	0.014



The test data consists of d = 2, 4, 6, 8, 12…***n***-dimension of real data (Iris Plants Database (UCI, 1998)) in integer format. For this particular project, we used a block size of eight (8) kb as the database storage block size; nevertheless, we used the same number of data items for all different dimensions. The heuristics ignore the potential hitches of the linear increase in the size of the database as a result of increase in dimension. It would have been more realistic though, to keep a constant data content size for the database. Thus, a better alternative and more accurate thing to do is the reduction of data rows in the database as data dimension increases so that the size of the database remains constant.

Hence, the main significance of our work is found in the unique technique for extending a node (to form a super node). From the discussions above, it is obvious that the ability to produce minimal, leaf centred, non-over flowing super-node shields the structure from early deterioration. The algorithm was implemented for 2D data; all our experiments were conducted using two-dimensional data for easy illustration. Nevertheless, the structure algorithms can work for any number of dimensions.

Table 4: Example dataset from iris (obtained from section 4.2.5 (A))

Sepal Length	Sepal Width	Species
5.3	7.8	setosa
5.1	3.8	setosa
6.9	3	virginica
5.4	3.5	setosa
5.1	3.3	setosa
5.4	3.9	setosa
7.4	2.9	virginica
6.1	2.8	versicolor
6.5	2.9	virginica
6	2.7	veriscolor
5.8	2.8	virginica
6.3	2.3	veriscolor
5.1	2.5	veriscolor
6.3	2.4	veriscolor
5.5	2.4	veriscolor
4.9	3.6	setosa
5.4	2.6	setosa
5.7	3.2	veriscolor
4.8	2.7	virginica
5.5	3.6	setosa
6.1	3.3	veriscolor
5.7	3.7	

Sheet1

Table 5: Example representation of x and y coordinate of the centroid of MBRs

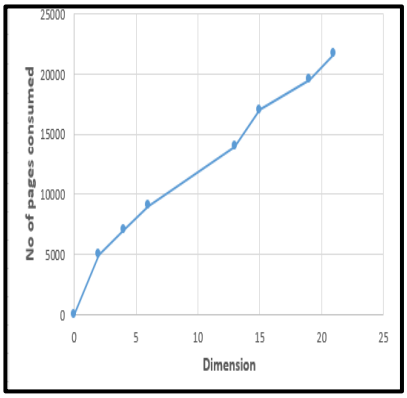
Obj_id	X (centroid)	Y (centroid)	Class
A	2.2	2.4	
B	2.5	2.7	
C	2.7	2.9	
D	2.8	3	
E	3.2	3.4	
F	3.5	3.7	
G	3.7	3.9	
H	3.2	2.5	
I	2.9	2.6	

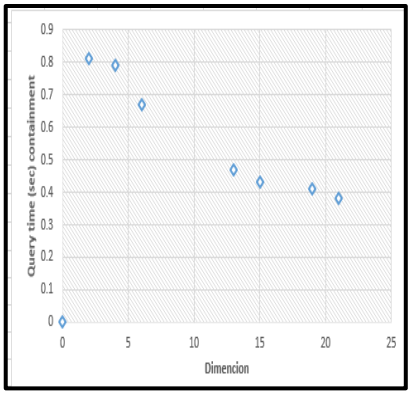
For illustrative purposes, we restricted our attention to 2-dimensional X-trees with varying branching factors, depending on the application and resource at hand. However, a branching factor of four (4), shows a great performance in the presentation of the concepts and extensions to higher branching factors (which fills a logical disk block) tends to affect the behaviour of the structure. It is significant to recall that the nodes referred are the pages on a computer storage (disk) therefore, constructing the tree, is expected to consider the fact that only a minimal number of disk pages are required to be accessed, before any query operation or the indexing structure is considered optimal.

B. Result

Table 6: Outcome of the proposed system

	A	B	C	D	E	F	G
1	Dimension	No of pages consumed	Size(mb)	Max per node (M)	Query time (sec) NN	Query time (sec) containment	#OF ROWS
2	0	0	0	0	0	0	0
3	2	5003	39.03	206	0.002	0.81	1033253
4	4	7045	54.95	146	0.002	0.79	1033253
5	6	9074	69.57	115	0.002	0.67	1033253
6	13	13995	109.16	74	0.002	0.47	1033253
7	15	17037	132.891	61	0.002	0.43	1048575
8	19	19561	152.578	53	0.002	0.41	1048575
9	21	21668	169.016	48	0.002	0.38	1048575





(a) (b) (c)

Figure 35: Performance of aX-tree on some real-world data (a) Number of pages consumed versus dimension on Real Point Data (b) query time versus dimension on Real Point Data.

C. Performance:

Modification1: The existing X-tree structure lets the size of a super-node be many times as large as the size of a standard node. Whereas, the X+-tree, permits the size of super-node to be at most the size of a normal node times a given user value MAX_X_SNODE. If the super-node grows bigger than the higher constraint, the super-node is divided into two (2) new nodes. These two (2) scenarios have the tendency to deteriorate tree performance as such, we have limited the size of the super node to just a size double (2times) the normal node size.

Modification 2:

In very high dimensional datasets, a linear arrangement of the internal nodes becomes more effective, owing to high overlap, thus, most of the internal nodes, if not the entire directory, have to be accessed anyway. Searching the entire directory may be useful because a linearly arranged directory requires a smaller amount of space and may be accessed much quicker from disk than a block-wise reading of the directory. To improve this behaviour, we have restricted the super node only at the leave level as such, yielding an increased speed because the block reading of the directory terminates at the index level

Modification 3:

Normally, increasing number or size of super nodes, will cause the height of the X-tree (that matches to the amount of page accesses essential for point queries) to decrease with growing dimension but in our case, we are already guaranteed that the height grows exponentially only at $\log_n(\mathbf{C})$ which improves the tree performance.

The Performance of any search algorithm, when the underlying data is indexed with a hierarchical structure (e.g. tree), is evaluated by the amount of reads (***disk access)*** required for finding the desired object(s) in the database. Thus, the branching factor of the structure is selected so that the size of a node equals the size of a disk block (or a many of it) or the size of page (file system).

In some applications involving a database with high-dimensional data, (Berchtold et al. 1996), NN queries are usually significant thus; the major concern for NN search for databases stored on a tree data structure is the ***CPU-time.***

D. Space and time complexity:

In terms of space and query time according to (Arya et al. 1994, Cazal et al. 2013), for dimensions higher than two (2), it is quite difficult to find algorithms that are worst case efficient. Nevertheless, for the **a**X-tree (our proposed structure), the capacity of every node is restricted to the value of the maximum number of entries (MBR or rectangles calculated in equation (**L**)), obtained by calculating the amount of available space spanning from 4kb page block size. Thus, for an optimal, efficient and cost effective I/O operation, every **a**X-tree internal node must occupy only a single disk page, giving us the number of pages at leave (or total leave nodes).

Data entry:

The algorithm takes as input an array of pointers to a set of rectangles for which the midpoints are calculated) or points and a description of maximum children count for each node (i.e. Maximum node entries).

4.2.6 Conclusion:

We discussed indexing of high dimensional dataset, based on spatial methods. This study shows that the X-tree has performed exceptionally well in terms of efficient query performance, based on spatial data indexing method. However, we found out that the X-tree starts to depreciate in performance and possibly degenerates to a linear structure when the dimension of data increases above a particular degree. Consequently, we have examined ways of improving the existing X-tree and thus propose (**aX-tree**) a method of building a packed x-tree by bulk loading the existing X-tree structure. The packed X-tree (**aX-tree**) performs better for dimension as it allows for the pre-processing of the data before loading into the tree structure. Packing or bulk loading methods of data insertion, constructs a tree at a time rather than entering each object into an empty tree one by one iteratively. A good bulk loading method builds quickly for static objects and ensures a reduced amount of wasted empty spaces on the tree pages. The **aX-tree** performs faster by loading the tree with all spatial objects at once, thereby reducing empty spaces in the nodes of the tree and producing better splitting of the spatial objects into nodes of the tree. Other characteristics of the **aX-tree** includes: a**) reduced** tree height, b**)** good quality (hierarchical as possible) directory node, **c)** minimum overlap and **d)** condensed area the MBR.

CHAPTER 5. PROJECTS ACCOMPLISHED and APPLICATION OF PROPOSED METHODS

5.1 Introduction:

This chapter reviews all the work we have done during the course of this research programme. We present all our finished/published work in this chapter. We described all accomplished projects briefly, with a link to their publication site. Currently, we are working on a project that integrates all the methods described in chapters 4 and 5. Our project sprang from the knowledge acquired from all the completed projects and from the materials that were studied. In all, this research work sets out to adjust, improve and implement new methods and algorithms for improved existing methods for large spatial or (non-spatial) data management. We implemented the core component of our research expedition and we tested the system using varying forms of data including real and synthetic data.

We highlighted only the vital details from all our published work and threw light to an ongoing project, which is like an implementation of all the systems we have proposed during the course of our investigations. In very recent times, large and small (both private and public) organizations have a heritage of operational databases (spatial or non-spatial databases), which are also somehow always linked to a spatial meaning. These organizations access several databases such as census, economic, security, image, multimedia, decision and policy making, statistical information for planning, industry, education, medicine, intelligence etc. The exceedingly large size of these data sets makes it difficult to search for meaningful patterns or relationships from within the datasets; this therefore presents a great challenge and difficulty in making useful organizational decisions/policies. SDM is the process of finding interesting and yet unknown, but theoretically useful non-trivial trends/patterns from large geographic or organizational datasets (spatial or not). By taking advantage of location information, SDM organises interesting realities. Thus, precise features of spatial data, which impede the use of general-purpose DM tools and techniques (algorithms, methods, models) include robust data types (e.g., extended spatial objects), implied spatial relationships among the variables, self-sufficient observations and spatial autocorrelation among the features (***Samson et al 2013)***. SDs are enhanced for storing and querying data, which stores objects based on their geometric value and space. Recent research on big data has majorly focused on spatial and temporal data (Vieira and Tsotras, 2013; Tian et al., 2015). It was established, according to the authors, that the reason for the attention on big data is simply that most of these kinds of data, have to monitor the behaviour and position of an object/event over

space and time. Notwithstanding, the wide and increasing availability of collected data (spatial or non- spatial) and the explosion in the amounts of spatial data produced daily by several devices such as space telescopes, smart phones, medical devices, and many others calls for specialized systems to handle big spatial data (**Samson et al., 2016; Samson and Lu 2016**;).

Regardless of huge effort in SDM research methodologies, most modern researches reveal that there still abound sensitive areas and matters of locality-time data management that still require tackling. **Mauder et al., (2015); Kamlesh et al., (2015); Billings (2013); Meng et al., (2014); Huang and He (2015; 2014); Kang et al., (2014); Moussalli et al., (2014; 2015); Secchi et al., (2015); Eldawy et al., (2015); Tian and Scholer (2015) h**ighlighted these problems to comprise uncertainty, scalability, locality privacy, inconsistency in handling S and traffic-aware navigation. These concerns according to them emanate from physical influences that often cause the locality-time data to be noisy and inaccurate. Consequently, data analysts are daily pursuing solutions to them. Moreover, as established in Li et al., (2016), most big data are spatially referenced; therefore, SDM remains the best strategy for evaluating these types of data. In this study, we focused on ways of improving existing systems and proposing novel techniques for mining data from spatial databases effectively. Large spatial datasets are easily characterised by four (4) major standards including ***languages***, ***index structures***, ***query processing***, and ***visualization***, however, we have focused majorly on ***indexing*** with a little touch on ***query processing and visualization***.

The work is organised as follows: in section ***5.2***, we carried out a detailed investigation on the background and basis SDs in the topic **"Spatial databases - an overview"** (--*book chapter – published*). This project covers a detailed introduction to the SD subject matter. This project looks into the fundamentals of SDs with a thorough description of their basic component, operations and architecture. In section ***5.3***, we discuss the topic "**paX-DBSCAN: a proposed algorithm for improved clustering**". The project (*book chapter – published*) is a proposed model for parallel clustering of large SD systems; it focuses mainly on the application of parallel computing methods and techniques for the bulk loading of the existing X-tree, in order to improve the performance of the DBSCAN clustering algorithm. A full description of how the system can be achieved was given in full details in this project. We present the discussion of another project that was accomplished, on this programme in section ***5.4***. "Large spatial datasets: Present Challenges, future opportunities". The project is a thorough investigation into the impending problems and available opportunities for large (spatial/non-spatial database) systems. It was published as a conference *paper*, in the conference proceedings. The article also made a note of the current trend in the area of SDM. The next project we carried out in this research work is described in section

5.5. It describes a new and improved algorithm for ***DBSCAN*** spatial clustering method. A *conference paper* titled **"Spatial Clustering in Large Databases Using Packed X-tree".** This project looks into how the ***DBSCAN*** algorithm works. Different from what we have in **paX-DBSCAN**, this project describes a new algorithm that improves the performance of the existing DBSCAN clustering algorithm using a packed X-tree and does not require the **minp and eps** values. Section ***5.6***, is an aggregate classic that produces an efficient technique for spatial modelling of human skin related pixels. This project combines all the work done during the course of this research. The model developed in this project is robust and versatile and can be useful in many other fields of machine learning or general recognition procedures. We applied the model to detection human faces in an image; the result is presented in Section ***5.7***. The project described in Section ***5.8***, discusses our main work in this research, geared towards an improved structure for indexing large and high dimensional datasets using spatial data models. The project produces the article titled "Large Spatial Database Indexing with **aX-tree**". The paper looked into a detailed outline of the main **aX-tree** packing techniques for static spatial databases that performs better in space management, through an effective packing algorithm. Every other algorithm in this work is based on this particular project. This improved structure produces two main advantages: (1) Reduction in the space overhead of the index structure. (2) Production of a better response time (the **aX-tree** index structure has a higher fan-out---i.e. size of leaf node, so the tree will always end up shorter).

Spatial data are obtained from various sources (scanned maps, satellite images, other similar images and scanned materials). In most cases, we digitise this input image into vector format (similar to our descriptions in sections 2.11 and 1.4) or graphic raster data. Fundamentally, Any form of geographical image (ecological, biological, physical, environmental..) with **coordinate points** or **location, can be seen or used as spatial data.** The tremendous increase in the volume of spatial data freely available for information processing in recent years has been overwhelming. This increase is evidently due to the advancement of telecommunication technologies, in addition, digital sensing devices has equally led to the need for a proper and efficient management technique for handling large pervasively available GIS data which normally come in very high dimensions. Regrettably, though GISs already support SDM, they do not possess the functionalities of a **DS** that directly support the spatial attributes (shape descriptions, area/extent, point coordinate) of spatial objects that are stored in a database. Thus, the development of spatial query **operators** is important. Geographical information systems (GIS) data is the core component of GIS. It is made up of two main types of data: Spatial data and Attributes data. Spatial data are normally used to

provide the visual illustration of a geographic space. They are stored as either **raster and vector types**. Therefore, this data is a blend of location and value data, which can be used for example for rendering a map. We build GIS systems to support **the manipulation of large geographic data** related operations such as **digitization**, **visualization** and **analysis**. However, these operations are not all SDM is about. There are other essential operations such as set operations (e.g. **intersection** between two spatial objects in a query --see Figure 5). Other database operations include "ranking" and "aggregation" (cannot be joined directly with GIS operations), are SDM issues that require specialised spatial procedure, techniques and methods.

Consequently, regardless of the advances, tools, techniques and products of GIS, commercial and research DS still have considerable need to integrate the support for spatial data in their systems. We have looked at **GISs** (example spatial databases), with application domains, where data normally has a spatial component with moderately low dimension (x, y, z...). However, there are still many other application areas with a significantly high dimensionality not necessarily spatial (example image databases). In these kinds of datasets, the data usually consists of made up other data (a collection of objects), and the high dimensionality comes as a direct product describing the objects through sets of features (section 3.3), otherwise known as a feature vector (section 3.3). Examples of features according to Samet (2006) include textures, colour moments, colour, shape descriptions, area/extent and so on.

In several applications, the databases are normally very large and consist of data objects with several (tens or a hundred of) dimensions. Thus, querying these databases requires the use of appropriate indexing techniques, which provides an effective access to high-dimensional data. Due to the high availability of large datasets, these applications essentially depend on large amounts of spatial data. In most cases, the trend for these databases is to map their multidimensional objects to **feature vectors** in a high-dimensional feature space and then perform queries against a database of those feature vectors. According to Chakrabarti and Mehrotra (1999) Feature-based similarity searches are emerging as an important search paradigm in DS and the technique used according to the authors is **to map the data items as points** into a high-dimensional feature space, which then indexed using a multidimensional data structure.

Similarly, according to Patrick et al., (2015), domain-specific objects (entities) present in many applications are easily compared and categorized, when they are represented as high-dimensional feature vectors. For detecting object similarities and quantifying clear groups, most analysts frequently visualize the vectors directly in order to identify **clusters.** Likewise, based on the elucidation in Samet (2006), Searching in high-dimensional spaces (easier with point and range

queries, because point and range queries do not involve the computation of distance) is time-consuming, when it comes to performing similarity queries. Regrettably, in high-dimensional spaces, many geometric data structures fail to work well (Cazal et al., 2013). Many multidimensional data structures for high-dimensional feature space indexing, perform poorly when the dimension increases to a certain level. Thus, we have examined most existing methods and techniques for managing the "pervasive" high-dimensional data spaces, which are often present in many application areas (medicine, engineering…), and we have proposed improved novel approaches for indexing large amounts of point and spatial data in high-dimensional space. What we present are new index and management structures, which considerably improves the performance of indexing high-dimensional data. The proposed structures are *multidimensional data structures for indexing high dimensional feature spaces, for pure data partitioning (DP) and indexing.*

Research has established that Indices used in managing multidimensional spatial databases perform effectively in low dimension e.g. when the dimensions (*d*) of a dataset (i.e. the attributes) is below 16 for large datasets. Nevertheless, the performance starts to degrade and degenerate for *d* greater than (>) twenty (20), to the level of sequential search (Berkhin, 2006). Hence, for d > 16 (i.e. with more than 16 attributes), we consider the dataset as high dimensional. The main motivation for this project is the need for improved ***speed, optimised storage utilization, and precision in information retrieval*** from large multidimensional spatial databases, using high dimensional (spatial/non-spatial) objects feature vectors. The R-tree index structure and its variation do not require point transformation, whilst storing spatial data (Berchtold et al., 2001). The structure also proves efficient for spatial clustering (a vital issue in the performance of tree based indexing structures). Nevertheless, they are not adequate for high-dimensional datasets because the **structures support high overlap of the bounding boxes in the directory/internal nodes**, which increases with increase in dimension. The drawback of this is that *most large DBs are often represented using high-dimension* ***feature vectors.*** So because **feature spaces often** tends to contain multiple instances of similar objects (Samet, 2006), the database built using such a feature space is bound to be clustered, therefore, if the database is indexed with an R-tree, there would be cases of redundant search of rectangles due to the high overlap between MBRs of the R-tree nodes. Several new index structures (A-tree, VA-tree, X-tree…), according to Mamoulis (2012) have been proposed. These structures outperform the R-tree for indexing high dimensional data but most of them degrade in performance when dimension increases (Berkhin, 2006; Berchtold et al., 2001; Mamoulis, 2012). Therefore, based on these grounds, we propose some improved algorithms mostly based on the logic of the existing ***X-tree.***

The aim of our project is to improve the performance of the existing algorithm for high dimensional spatial datasets and large/ big data in general. In addition, our work also suggests new methods, which are geared towards the improvement of information retrieval processes in large SD. In this chapter, we have listed some research work that we have already completed and published. As we have discussed above, the whole essence of our investigations is towards a better system for optimizing the performance of large (spatial /non-spatial databases).

5.2 Spatial databases - an overview:

This work in this project strives towards a general survey of **SDBS**. The full details are found in *Samson et al. (2017[b])*. We carried out a detailed investigation on the background and basis of SDs in the topic **"Spatial databases - an overview"** (--*book chapter – published*). This project covers a detailed introduction to the SD subject matter. This project looks into the fundamentals of SDs with a thorough description of their basic component, operations and architecture

A. Description:

A SDBS is a DS that deals with spatial and/or temporal data types in their query language and data model. They normally support spatial data types in their implementation, providing at least spatial indexing and efficient algorithms for spatial join. The attributes for referencing the spatial or geographic quantities of an object in SDs, allows the objects assume a two (2) or three (3) dimensional position in space. This project looks into the basics of SDs and describes their basic operations, components and architecture. The study pays major attention to the description of the *query Language, data models, processing queries, query optimization* and *indexes* of a SD that approves them as a necessary mechanism for data storage and data retrieval from a multidimensional database of high dimensional spaces.

B. Method:

The method adopted in this project, entails a detailed analysis covering spatial SDMS architecture, and data representation. We presented a thorough explanation of the different SDT (polygons, lines, points or region) and how they are stored, manipulated, managed and converted. The whole procedure from storage to manipulation takes two (2) major stages (listed in section 5.1.1) within the database.

5.1.1 The basic steps to converting from polygon to point is itemised as follows (see algorithms below):

1. Build a two (2)-dimensional MBR (bounding box) around the points by:
    - Determine the convex hull of the set points or regions see Figure 36
    - Reject redundant points as regards the solution at hand, and
    - Build the rectangle/polygon (from only convex points)
        - The MBR contains the rectangle's ***m-number*** of vertices. *See section 2.7.1.*

2. After the steps above
    - Estimate the centroid/centre of the object (on the x-axis using the formula in 4.2.4) from the rectangle (MBRs).
    - Build the relation with the point coordinates as described in section 2.10.4 (in most cases, this is used as the point for representing the object, rather than the object extent or shape).

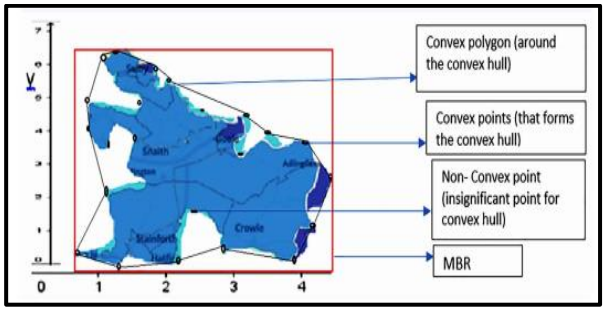


Figure 36: Example graphic of stages in polygon conversion, for spatial objects (objects with extent)

Similarly, we looked at many other SDBMs related issues including *information retrieval*, ***spatial indexing*** and extension of classical relational/classical database for accommodating spatial datasets. The extension consists of optimizing the models, improving the choice of ***DBMS*** for building spatial databases, optimizing spatial query processing etc.).

Other topics discussed include components of a spatial database, mechanism and techniques of managing/handling high dimensionality in spatial databases and many others.

C. Result:

Many algorithms were proposed in the course of this project and we have presented some of them Figure 37 and Figure 38.

1. *Find a two-dimensional bounding box (MBR) around the points by*

Assuming all the points in p are distinct
Let the space be of dimension 2
Then
{
Enter coordinates representing each point (2 P),*
Set of all points = p
Set of all convex points = C

//Where $p = (a, b, c \ldots . r)$
Then for all $p_i \in p, (i = 1 \ldots . r)$
Find $\{s \mid s \leq p \wedge p \in C\}$

Start with the s value that has the smallest y-coordinate.
Sort the points following a polar angle with s to get simple (convex) polygon.
Consider points in anticlockwise order, and eliminate those that would create a clockwise turn.
}
Return convex polygon
// sorted list of points t along the boundary (moving anticlockwise) with the convex hull of S

(a)

Given:
A set of convex points say p // the points constitutes the convex polygon
Produce:
The minimum bounding box for the set of points say q// the output points is expected to be smaller than input points

Start:
// let V be the number of edges on the convex polygon
For v = 1...V
Draw a rectangle with v_i *as the base (i.e.* v_i *is collinear with an edge of the rectangle)*
Calculate the area A of the rectangle
Repeat for all v
Display the rectangle with the smallest area as the minimum bounding box (mbb)

Other simpler method can be used by just finding the xmin,ymin,xmax,ymax values

Start
Get min x, max x
Get min y, max y
Using these bounds
Draw the rectangle around the points

(b)

Figure 37: (a) Algorithm for Finding the Convex hull and constructing a minimum bounding box (MBR) (b) Algorithm for building a convex hull around a Polygon

2. *Estimate the Centre (centroid) of the object (on the x-axis) using the MBRs.*

$$C_{x,y} = \left(\frac{x2 - x1, y2 - y1}{2}\right) \ldots\ldots\ldots \text{Equation 1 (for a simple rectangle)}$$

Generally, this equation can be broken down into the sequence below.

i. *Get the* **x** *and* **y** *coordinates of each vertex v, in any order*
ii. *For each* $v_j \in V$ *of the* i_{th} *coordinates (***i** *=1,2... d) in a* **d** *dimensional space*
iii. *Compute centroid:*

$$C_i = \left(\frac{dxi, dyi}{d}\right) \ldots\ldots \text{Equation 2}$$

d is the number of dimension

iv. *Return a set of coordinates* **i**, *(e.g. x, y for just xy coordinates)*

Figure 38: Algorithm for estimating the Centre (centroid) of on object

5.3 A novel algorithm for parallel clustering (paX-DBSCAN)

This algorithm is an improvement of the work we did in the first year; however, a parallel method of managing large datasets was investigated in this case. The published material with the full details is found at ***Samson and Lu (2016).*** The project (*book chapter – published*) is a proposed model for parallel clustering of large SD systems; it focuses mainly on the application of parallel computing methods and techniques for the bulk loading of the existing X-tree, in order to improve the performance of the DBSCAN clustering algorithm. A full description of how the system can be achieved was given in full details in this project.

A. Description:

In this project, we applied a parallel technique to the bulk loading of the existing X-tree, in order to improve the performance of the DBSCAN clustering algorithm. A full description of how the system is archived is presented. In this work, we looked at another algorithm to extend the X-tree spatial indexing structure. We assumed that the environment is composed of a number of processors based on a shared-nothing architecture, where each processor manages its own disk(s) and main memory. We have also assumed that there would be no re-organization of the data taking place after the completion of the index construction process, that is to say, the data remain assigned to the same processor. This approach assures a load –balance, during the process of index construction.

B. Method:

In this section, we describe the method for achieving our proposed system (***paX-DBSCAN clustering algorithm***) in full detail. The approach we adopted for parallelizing the DBSCAN algorithm, by implementing our proposed **aX-tree** is very simple (see figure 39) and involves the following simple logical steps below:

Let **R** be the given large spatial dataset

1. Store the data from **R** in a parallel spatial database
2. Construct **aX-tree** index on the database
3. Run our DBSCAN algorithm for clustering purpose
4. Integrate the result from all the processors, to get the final output.

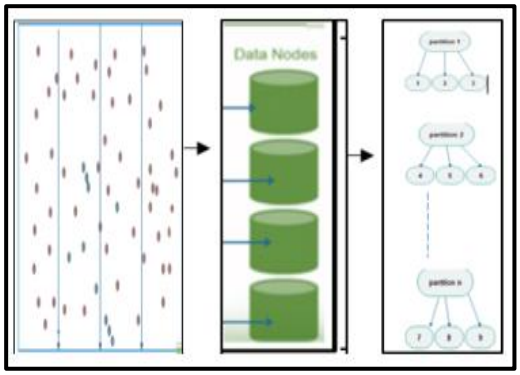


Figure 39: Partitioning technique for our proposed parallel aX-tree

Our aim is to run the DBSCAN algorithm on machines (that are located at distant sites) individually with each of the client nodes producing its own local cluster. The clusters from the various sites are then returned to the master node (***Nm***). On the master node, we construct a global cluster, which synchronises the entire local clusters. The master node (***Nm***) assumes the job by aggregating them into one result for the final output cluster. The algorithm, parallel adjusted X-tree – DBSCAN (**paX-DBSCAN)** --another novel parallel version of the existing DBSCAN clustering algorithm, is presented and described in detail. The method is applied to a distributed computing environment, where each node (computing site) is a sub-node or branch of the **aX-tree** spatial indexing structure. Different from all our existing methods, the algorithm is enhanced by implementing the **aX-tree**. Apart from the indexing structure (**aX-tree**), we also proposed a new procedure for the DBSCAN, which does not hinge on the values of the ***Eps*** (the main factor causing delayed computational time of the original DBSCAN algorithm).

C. Result:

Technically speaking, our investigation led to the development of various algorithms (presented below). The algorithms highlight the approach we have adopted for parallelizing the DBSCAN by implementing the ***aX-tree***. We implemented the DBSCAN algorithm by applying it on machines (that are located at distant sites) individually with each of the client nodes producing its own local cluster. On each client node (Cn_i - *for* i_s *= 1 to the total number of n clients*).

Procedure 1: statement of the problem

Initial Problem Statement

Given:

A set of points (*n-dimensional*) in a database say **P** such that **P = {P1, P2,....,Pn}**

A set of computers ***N*** such that ***N = {N1, N2,....,Nn}*** connected via a high performance computing infrastructural network

Find the clusters (density-based) which obeys a given ***Eps*** and ***MinPts*** constraint.

Procedure 2: the STR algorithm

Sort-Tile-recursive Pseudocode:

P= the count of high dimensional objects in a 2d Cartesian plane.

Let ***N*** = the total number of available computer.

Let **m** = the maximum capacity of a node (number of node entries that can fit leaf or non-leaf node).

Let **n** = dimension

// J = P / m = the estimated total number of leaves required.

Step 1: by using the x-coordinate as a key; sort the objects (rectangles) based on the x-coordinate of their centre.

Step 2: Determine the maximum node entries.

Step 3: Order the sorted rectangle into **J = ⌈ P / m ⌉**.

Step 4: Divide the sorted rectangles into **r** groups of vertical slices.

- For two dimensions **r** = $\sqrt{\mathbf{J}}$.
- For dimensions more than two, let **p**= dimension, **r** = $J^{1/p}$.

step5: Sort the new group r groups again based on y –coordinate of the rectangles centre into

Output:

Procedure 3: the clustering steps

Basic clustering steps:

Divide the input (P) into **r** partitions such that **k = k1, k2…, kr** and distribute these partitions to the ***N*** available computers.

Run the proposed DBSCAN clustering algorithm in each partitions concurrently

Finally ***combine or merge*** the clusters from the partitions into a global cluster for the entire database.

Procedure 4: general pre-processing steps

Start:

1. **Take** a sample of data from the large dataset
   // the sample can be chosen as a percentage (1, 2 or any percentage of the data, though 1 is a good choice) of the given dataset for point data **P**, but in the case of spatial object (objects with extent), we could convert shapes (lines, regions, areas) *to points* by obtaining their mid-points.
2. **Find** the centroid of the complex shapes (regions, rectangles, lines etc.), from the sample using the simple equation below. Note the formula considers the bounding rectangle of the spatial **object** only.

   $$C_{x,y} = \left(\frac{x2 - x1}{2}, \frac{y2 - y1}{2}\right)$$

   //In other cases, getting the centroid on a polygon based on the number of **j**-vertices will generally require a different formula.
3. **Calculate r.**
4. **Divide** the sample space into **r** vertical slices.
5. **Bulk load the aX-tree into main memory of the server**
   // the extended node (***super-node***) is applied only in the first level (the bottom of the tree) to avoid the problems of hyper rectangles overlap.
   Step 1: by using the x-coordinate as a key; sort the objects (rectangles) based on the x-coordinate of their centre for complex objects.
   Step 2: Sort the new group r groups again based on y -coordinate of the rectangles centre into.
6. **Output:**
   After loading the **r** groups of rectangles into nodes (**pages**) the output = (MBR, Node Id) for each leaf level node that loaded into a temporary file to be processed in phase two of the aX-tree algorithm.

Procedure 5: partitioning---figure 39

Start

// Using the boundaries of each leaf node as the boundary of the MBR of the leave nodes, then scan the data in parallel and assign each record to its overlapping partition.

i. **Store** the location of each partition (***gl***) and their range in a file in the server in the form (O_{Id}, ***MBR***, N_{Id}) and call it N_k. (**i.e. k** partition in **N** computer)
   // O_{Id} is a (child) pointer to a lower level node.
   // ***MBR*** is the he rectangle enclosing it (which covers all regions or points in child node).
   //N_{Id} ***identifies*** the partition (computing node) where the object is stored.
ii. **Partition** the data into the ***N*** number of computers by computing **r** and mapping out its horizontal range for partitions k_i. **(i = 1….r).**
iii. **For** each new entry point (from the new dataset) the server compares the range its spatial attribute to the range of the **MBR of the global leaf** (***gl***), and sends it to the right node partition as shown in figure 18.

Step iv is repeated until the entire dataset has been fully distributed.

5.4 Challenges and future opportunities for large spatial datasets:

A. Description:

We present the discussion of the project "Large spatial datasets: Present Challenges, future opportunities". The project is a thorough investigation into the impending problems and available opportunities for large (spatial/non-spatial database) systems. It was published as a conference *paper*, in the conference proceedings. The article also made a note of the current trend in the area of SDM. Detailed in ***Samson et al., (2016)***, the project investigated the issues regarding large

spatial datasets (*architecture, features, design technologies and access methods)*. We also looked at possible ways to augment the operational behaviour of existing algorithms for managing large spatial datasets. The study shows that storage utilization is the main challenge combating effectual management of large spatial datasets. The other factor is computational complexity. These two factors are distinguished by the size of the data (spatial big data) that inclines to exceeding the capability of frequently used spatial computing systems, owing to their *variety, volume and velocity*). Fortunately, this project has highlighted all the various ways of combating the above problems, using functional programming methods or parallelization techniques.

C. Method:

We discussed to a large extent, the characteristics and sources of large spatial datasets including differentiating between the terms *"large*" and "*multidimensional*". We expatiate on the challenge of big S, which majorly arises from its ***complexity (variability)***, ***size (volume),*** and ***rate of growth (velocity).*** According to Bhosale and Gadekar (2014), the three (3) mentioned above, make a given dataset complex to be gathered, processed, maintained or analysed by current technologies, methods and tools. Likewise, we discussed the architecture and design technologies and access methods required to manage large spatial datasets successfully. As part of this aspect of the subject, we elucidated on the different areas to be considered in the management of large spatial dataset; including representation of S (the Raster and vector data types—discussed also in sections 2.11 and 3.3), multidimensional data structures**,** major design issues (which includes *Indexing, Language, Visualization and Query Processing for large multidimensional datasets*). Finally, we looked into the present challenges of large database systems with emphasis on *storage optimization and speed. Subsequently*, we presented the various opportunities that can boost the research in the area of spatial database management, which covers areas such as Cloud Computing, Spark technology, Bulk-loading, and **Parallelization**, all of which greatly benefits our underlying research area.

D. Result:

Following our investigations, we have recommended potential ways of enhancing the management of large spatial datasets below

1. Develop a novel indexing (spatial) structures and methods, for handling the complex nature of real-time analytics, for which geographical data is typical of.
2. Develop a mechanism that explores explanatory and unstructured relationships.
3. Design competent methods for visualizing data
4. Design new methodologies to combat error propagation

D. Research Significance:

Like other projects we worked on, this work has also contributed to some extent to the research community.

5.5 Packed X-tree: an improved spatial clustering method for in large databases

A. Description:

In this section**,** we describe our next project. Also carried out in the process of this research. The project describes a new and improved algorithm for ***DBSCAN*** spatial clustering method. *The project was published as a conference paper* titled **"Spatial Clustering in Large Databases Using Packed X-tree".** This project looks into how the ***DBSCAN*** algorithm works. Different from the system in **paX-DBSCAN**, which is a parallel version; this project describes a novel algorithm that enhances the performance of the existing DBSCAN clustering algorithm by applying a packed X-tree and does not involve the ***eps and minp*** values (the two important values/parameter that reduces the performance of the existing DBSCAN algorithm). In this project detailed in ***Samson and Lu 2018***, we have broadly elucidated how to achieve the proposed system and we have additionally suggested a new efficient method for ***finding the k-NN of spatial objects*** (which is the main highlight of the proposed system) in a big database. Our study reveals that the method proposed in this work is very competent and will greatly speed up the execution procedures of density-based clustering in big datasets as compared to existing methods. The proposed system is an enhancement on the work we described in ***Samson and Lu (2016)*** and ***Samson et al 2017.*** Notwithstanding, new suggestions for enhancing the performance of the proposed algorithm is made.

B. Method:

In this project, we illustrated how the existing *DBSCAN* clustering algorithm works. We went further to describe the two important parameters (***minp*** *and eps value*) which describes the algorithm. Furthermore, we described how various index structures have been applied to hasten up the performance of the algorithm. Finally, we stated the major limitations of this algorithm (which majorly include the certainty that the R-tree and its variations are not sufficient for large high-dimensional datasets as the indexing structure permits high overlap of the MBRs in the directory that increases with increasing dimension) based on our findings. Consequently, we presented our new algorithm, as shown below. The highlight of this project is the ability to

automatically detect or decide the value of ***k***, which represents the two important parameters used in the existing algorithm. This value is obtain using the very simple heuristics below:

Finding K:

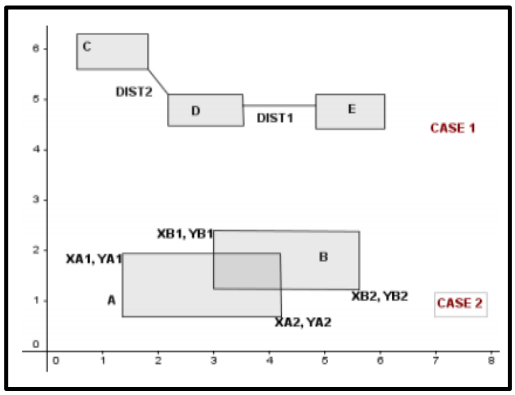


Figure 40: Relationship between the bounding boxes (rectangles), for finding KNN

Finding the closest rectangle or nearest neighbours becomes easy, using the **aX-tree** structure. This is so because the database is indexed or stored using the packed X-tree. The rectangles are already packed (bulk-loaded) into buckets (***MBRs***). Thus, based on this improvement, and because the MBRs are axis-aligned and packed in an organised or sorted order, the only thing we to do then is:

Since we already know the *max and min* values of the XY coordinates for ***rectangle A* in Figure 40 as**,

R= (XA1, YA1, XA2, and YA2).

Therefore, if the right and left edges of the MBRs are *different that is.*

XB1 > XA2, XB2 < XA1

Then

We can compute ***Dis*** *(rD, rE), as the distance (****Dis****) between rectangle D and E, where rD is rectangle D and rE is rectangle E.*

Else

For ***case*** → *YB1 < YA2 or YB2 > YA1*

Then

compute Dis *(rC, rD).*

Depending on the total number of boundary points around individual rectangles, depending on the dimension, a series of tests can be carried out. For individual vertices in the neighbouring (or adjacent) polygon, we now calculate the distance (using their Euclidean distance ***Dis***), to the individual vertex in the query MBR. Eventually, we compute the closest polygons in the study space.

Furthermore,

Given any two rectangles in k-dimensional Euclidean space containing

rectangle A (rA) and rectangle B (rB) as seen in Figure 40.

Let rA = (Xa, Ya), and rB = (Xb, Yb), be their upper right and lower left corners respectively,

Then

The distance between *rectangle A and rectangle B is given as:*

$$\boldsymbol{Dis}\ (rA, rB) = min\{min\{dist\ (r_a, r_b)\}\},\ \forall r_a \in rA;\ r_b \in rB \qquad \textbf{(K)}$$

I.e., for any object or point say r_a in rA, find the smallest distance to any object or point in rB (i.e. r_b); then find and store all the smallest distance in r_a.

After calculating these distances, then, for any object or point $\boldsymbol{r_a}$ (an object in rectangle A), find the **least distance** (min-dis) to any object or point in ***rB*** (mbr B); then find and store the least distance in $\boldsymbol{r_a}$. sort the objects or points in an ascending order.

Afterwards,

1. Construct a buffer for the best **k** closest neighbours.
2. Prune the MBRs based on the distance of the furthest from **object in position k**

C. Result:

In Figure 41, some of the algorithms developed in this project

Start from root
Locate qx-tree leaf node with lowest ***x-value***
→ ***L = 1***
→ While ***L ≥ 1***
1. Determine Φ > 0
/ **Φ** is the parameter that determines which cluster is dense*/*
/ value of **Φ** = **(at least) m +1** so as to avoid clusters that has only one object */*
2. Find** a neighbourhood value **(E > 0)
/ the value of **E** (nearest neighbour to an object s∈S) outlined in section D.*
3. Clustering
*i. Find **Bi** = {s ∈S: d(si, s) ≤ E }*
*// (**d** = distance between s and si for **all** i = 1, 2….m)*
*ii. If | **Bi** | ≤ **Φ**, **THEN***
*iii. **REJECT** B_i*
// B_i is an outlier
ELSE
iv. Find the relationship ($B_i \cup B_j \neq \emptyset$)
*v. **Repeat iv** until there is no more union*

(a)

Starting from the root node
→*Take the leftmost internal node, search all its child until you get to the leaf (i)*
While on the level → leaf
→ Start from the leftmost node
→Find E (k -neighbourhood as described above)
Begin clustering
→Find all directly reachable nodes (from the aX-tree) as section V (D)
→ Follow the rest of the Clustering (aX-DBSCAN AGORITHM) until the last leave node (r) in the current neighbourhood
→ Move to **the children of the next consecutive i +1 level of the tree and then continues until the last node in the tree is reached.**

(b)

Figure 41: *(a) Our clustering (aX-DBSCAN) algorithm* *(b) Performing our clustering algorithm*

5.6 Novel multidimensional spatial model for fast and accurate human skin detection

5.6.1 Introduction:

Skin detection algorithms can benefit greatly and perform optimally, if enhanced by spatial index and spatial access methods. Many colour-based systems have been modelled which performs well in this dimension, However, research shows that these systems suffer from certain major setbacks, including, individual pixel operation (otherwise known as Pixel-by-pixel or pixel-wise operation), high rate of false hit and poor performance in terms of predicting darker complexioned skin. First, for the darker skin colour problem, we propose an improved RGB/HS colour threshold. Secondly, for the Pixel-by-pixel problem, commonly used structures for performance enhancement in this regard is the quad-tree. However, the idea of repeated deep quad-tree-like tedious partitioning seems to be cumbersome and, in some cases, quad-trees have proven to have poor shape analysis and poor performance on pattern recognition due to their inability to compare two images with different translation or rotation efficiently, especially with an image that has different colours for every pixel, thereby losing effect. In this paper, we have proposed a new fast algorithm based on an improved, combined colour model (HSL and RGB) threshold value, for human skin/face detection from coloured 2D images using a packed maintained k-dimensional tree *(Pmkd-tree*). We compared the proposed system to traditional pixel-by-pixel procedures and we also implemented a novel packed quad-tree (Pquad-Tree), to speed up the quad-tree performance; nevertheless, the result shows that our proposed spatial structure performs better with very low false positive/negative rate.

A. Main claim of the paper:

There **two (2)** major contributions.

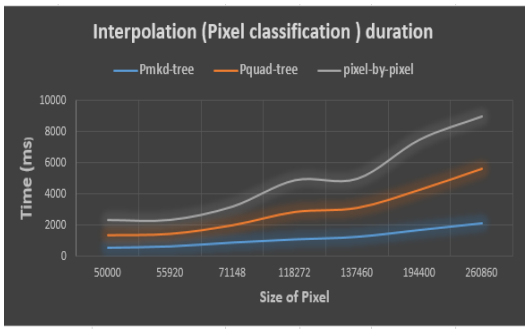
First, we proposed a new colour threshold for human skin classification, which is enhanced by certain constraints, making it more efficient as compared to closely related state-of the-art colour schemes (figure 42).

(a) (a-i) (b) (b-i)

Figure 42: (a) result of other colour threshold (b) result of our colour threshold

Kolkur et al. (2017); Jati and Selvam, (2008); Peer and Solina, (1999); Baskan et al., (2002) and Ali et al. (2018), suggested closely related colour scheme, but as you can see in figure 1(a), there scheme is limited by high false hits (positive and negative). Therefore, we added certain constraints on one or more of the colour channels (see result in figure 1(b)), to reduce the effects of the negative hits.

Secondly, we propose a novel spatial structure **P*mkd-tree*** (**Packed maintained k-dimensional tree**), which main purpose is to eradicate the common state-of-the-art ***pixel-by-pixel*** approach of pixel classification on a given image (as we applied in Figure 42 (a-i) and Figure 42 (b-i)) based on interpolation. To the best of our knowledge, this is the first time such structure is developed. The structure shows high prospect, in terms of speed (Figure 43(a)), low rate of false hits (Figure 43 (b)), reduced computational cost and complexity (Figure 43(b)) and high accuracy and precision rate (Figure 43(c)).



(a)

Method	Total No of pixel in image	Skin pixel detected	Time (ms)	Non - Skin pixel present	True Positive	False Positive	True Negative	False Negative	Precision / Accuracy (%)
Pixel-by-Pixel	50,000	20,510	1173	29,490	18,926	1584	29,490	0	92.37 / 96.83
Pquad-Tree	50,000	18,000	803	27,468	18,000	0	27,648	926	100 / 98.01
Pmkd-tree	50,000	18,720	473	32,064	18,720	0	31,074	206	100 / 99.59
Ground Truth	50,000	18,926	648	31,074	-	-	-	-	-

(c)

Only ≃20% of the pixels in an image is required to classify the pixels in a skin detection procedure

(b)

(d)

Figure 43: (a) time consumed on pixel classification based on the pixel-by-pixel approach, our approach and quad-tree approach. (b) The summary and main claim of our model (c) table showing accuracy and precision rate of our model on N-total amount of pixels. (d) A smoothed version of the image on figure 1 (a-i) and (b-i)), using our proposed spatial model and improved colour threshold.

B. Significance of proposed system:

It is true that many tree-based systems have been proposed for improvement of the skin prediction procedures, however, as discovered in this study, these systems are still faced with a high rate of false hits as shown in figure 2 (a), quad tree-based system, the common system used for this purpose is limited in speed by the deep quad-like partitioning strategy. Therefore, the effectiveness of our model as depicted in figure 2, will contribute to improving time and computational complexity in learning systems.

Though we have only applied our proposed model to the classification, prediction and recognition of human skin pixel in an image (image processing, pattern recognition, information retrieval and computer vision), this model is robust and versatile and is useful in many other fields of machine learning procedures including clustering (e.g. clustering skin pixels on the face as a blob in face recognition, cell/DNA clustering in biology for matching purpose etc.), design of discovery systems (e.g. gene pattern discovery and identification in bioinformatics, data mining and knowledge discovery etc.).

The main claim of our work is the summary in Figure 43 (b) and it says:

→ only ≅20% of the pixels in an image is required to classify the pixels, in a skin detection procedure.

The above proposition is true for all mage types and size, as long as **the defined skin colour threshold** is accurate. This can be proved by the results in Figure 43 and Table 7 of our work.

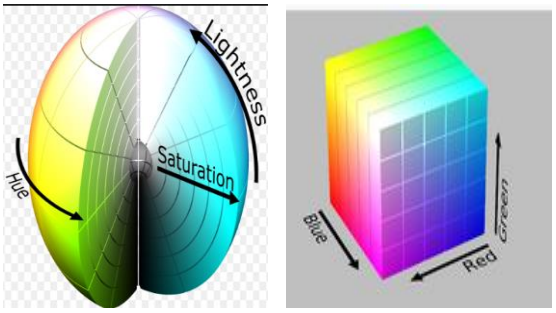
C. Elucidation

Human skin segmentation/detection is still an unsolved problem (Faria and Hirata, 2018). According to Buolamwini and Gebru (2018), there is a substantial disparity in the accuracy of classifying darker skin colours as against their lighter counterparts, therefore, requiring urgent attention of commercial companies in building genuinely fair, transparent and accountable skin analysis algorithms. Skin detection algorithms suggest the presence of a human skin in a digital image. It is an important preprocessing step for techniques like face detection and semantic filtering of web contents. According to Albiol et al. (2001), every colour space contains an optimal skin detector scheme such that the performance of all the skin detectors scheme is the same. In Patil and Patil (2012), the basic steps in skin detection include; representation of image pixels in colour spaces, suitable distribution of skin and non-skin pixels, skin colour modelling (which uses an underlying skin colour distribution characteristic on a colour space to detect, skin colour pixels quickly). Nevertheless, Skin appearance in images is affected by various factors such as illumination,

background, camera characteristics, and **ethnicity**, as such; skin detection using colour information can be a challenging task Kakumanu et al. (2007). Numerous techniques exist in literature for skin detection using colour, however, due to real-world conditions such as illumination and viewing conditions, many of these works are limited in performance. These techniques Tan et al. (2012), are prone to false skin detection in most cases therefore, not able to cope with the variety of human skin colours across different **ethnic groups**. Thus, in this paper, we have proposed a fast algorithm based on an improved, combined colour model (HS and RGB) threshold value, for human skin/face detection from coloured 2D images using a maintained ***Pmkd-tree*** (**Packed maintained k-dimensional tree**). Exhaustive channel toning facilitated enhancement on insensitivity due to luminance. The algorithm starts by compressing image quality and reducing the size of the image, thereby achieving only less than **60%** of the image size and quality, this pre-processing technique increases the speed of the application. Experiments show **that the proposed algorithm is characterised by very high accuracy rate, precision and efficiency** (See table 7).

5.6.2 Description of colour spaces (channels):

According to Phung, et al. (2002); Albiol, et al. (2001) and Kolkur et al. (2017), The *RGB (Red, Green, and Blue), HSV (Hue, Saturation, and Value), HSL (Hue, Saturation, Lightness)* and *YCbCr (Luminance, Chrominance)* colour models are the three main parameters for identifying and recognizing a skin pixel.



(a) (b)

Figure 44: (a) HSL colour model, (b) RGB colour model

In Nishad, (2013), the HS colour hexagon was described as what picture windows use in their colour picker to display the brightest possible versions of all possible colours, based on their hue and saturation. This justifies our decision to adapt the colour model as a choice tool for skin colour detection. Additionally, the characterization of colour range for skin detection is achieved by

manipulating the H channel of the HS colour model (Oliveira and Conci, 2009). From the RGB coordinates of the image, the values for H, S and L, are derived. The H channel of the HSL is applied to characterize the colour range for skin detection. The S channel defines the saturation of the H pigment. The L channel normalises the shade or saturation of both H and S.

The major difference between our work and these existing models in terms of colour, is the adaptability of the hue channel to different ethnic skin colour shades achieved through significant range normalization between these colour categories and the speed up of classification procedures using spatial models.

5.6.3 Related work:

Facial recognition technologies according to Simonite (2018) has proven to work less accurately on people of colour. One reason this may be so according to a study by Buolamwini and Gebru (2018), is that skin type classification systems are overwhelmingly designed to favour lighter-skinned subjects. With an error rate of up to 34.7%, leading to higher overall accuracy rates for identifying men of colour than women, their study, established that darker-skinned females are the most misclassified group. Illumination, pose, noise and expressions, are some of the opposing factors during face capture and analysis. These, according to Thakkar, (2018) and Tan et al. (2012), greatly affect the performance of facial recognition systems. Thus, among all biometric systems, according to the authors, facial recognition has shown the highest false acceptance and rejection rates. This means that face recognition systems will benefit if skin detection algorithms improve, since skin detection is a pre-processing step for image detection/recognition. Notwithstanding, most existing skin detection methods (Sun 2010; ban et al., 2014; Zortea, 2017), depend on building an n-dimensional histogram for pixel classification. In most cases, two (2) histograms are constructed at the start, with the training pixels, for training purpose. One histogram for skin related pixels the other for non-skin related pixels. Using these two histograms, a classification rule such as Bayes (or any other), is applied to **each** pixel of the test pixels or image in order to complete the detection process. One thing is common amongst these methods and approaches, **that is, individual pixel (pixel-by-pixel) examination for the purpose of developing the classification rule.**

i. Pixel-by-pixel operations (pixel-wise)

Many state-of-the-art techniques for human skin detection from images, depends wholly on pixel-wise operation. However, efficacy of the pixel-wise classification is limited Kawulok et al. (2014). The main objective of pixel-wise skin colour detection according to Xu et al. (2013) is to build a decision rule that classifies each pixel as skin or non-skin individually. Nguyen-Trang, T. (2018) and Kawulok et al. (2014), claimed that the performance of skin detection algorithms has not really been high due to high overlapped degree between "skin" and "non-skin" pixels. As a solution, they applied Bayesian classifier and connected component algorithms to identify individual "true skin" pixels using the first posterior probability threshold. Though this method helps to improve skin classification performance, especially the false positive rate, it has to go through the rigorous task of **checking all pixels individually, thereby not speed efficient.** Several state-of-the-art methods for skin detection use **single colour region approach** according to Hassan, et al. (2017), in contrast to this, they applied genetic algorithms, to determining optimal skin colour regions from a selected colour space, which considers skin colour as a union of multiple smaller CbCr colour regions rather than the aforementioned **single colour** region counterpart. However, even though they applied an optimization on the CbCr colour model, which they used in their work, **the image pixels were as other systems tested individually**. The work in Mortazavi and Ebadati (2019) similar to our proposed system, starts by reducing the size of the given image, and then applies RGB and YCbCr colour models. However, it eventually ends up with processing **individual candidate pixels** that are in the range of skin colour, for detecting human skin. Like other models presented so far, Omer et. al. (2018) offered a similar method of **individual pixel examination** using only the HSV colour model, but applied two different types of noise filters NOGIE (Noise Object Global Image Enhancement) and NOWGIE (Noise Object with Global Image Enhancement) for improved result. Other similar methods include; Bush et al. (2018), adaptive neuro fuzzy inference system (ANFIS) for skin/non-skin pixels detection. Kolkur et al. (2017), a new threshold based on a combination of RGB, HSV and YCbCr values for skin/non-skin pixels detection. Modified likelihood ratio, in addition to multi-scale, was used for classification in Roheda (2017) for **pixel-by-pixel** skin pigment classification. Likewise, by establishing some correlation rules between the chrominance components PCr and PCb of **a pixel P**, (Faria and Hirata, 2018) formulated two equations ($PCr - PCb \geq IP$ and $|PCb - PCbs| \leq JP$), must be true, to determine if a **single pixel** is skin or not. Note: IP=max between values (PCr, PCb) and JP=maxDistance between points ((PY, PCb), (PY, PCbs)). Vezhnevets et al (2003, September), also presented a similar system. In Kawulok et al. (2014), a spatial based system for skin detection was presented. The system applies a discriminative feature space as a domain for spatial analysis of skin pixels, based on textural

features extracted from some skin colour probability maps. The texture is extracted in the form of **seed,** taking advantage of the fact that real skin areas have pixels with high-skin probability threshold, if an image is binarized. A distance function is applied for finding the shortest routes from large blobs of the **seed** to every pixel. Eventually pixels not close to any of the seed blobs are rejected and then the skin regions are extracted. While this method seems promising due to its texture-based idea, the method does not involve actual space partitioning, which is typical of hierarchical classification, for efficient distance threshold queries. As such, query performance is impaired due to bulky distance calculation between seeds and individual pixels.

Ii. Tree based solutions

Dastane et al. (2018), in addition to Deep Neural Network and Naïve Bayesian model presented a decision tree-based solution for skin detection that overcomes the challenge of colour range thresholding. By calculating the probability of each pixel, their proposed equation is tested ***pixel-wise*** before a pixel is classified using a skin/no-skin decision tree. Their method performs reasonably; however, it still portrays the shortcomings of most existing systems including inefficiency with time due to stages involved in processing individual pixels and inaccuracy in certain skin type prediction, as stated by the author. The quad-tree structure in Roheda (2017), differ greatly to what we presented here in that for each 32x32 sized neighbourhood pixels, if a boundary is detected, the 32x32 neighbourhood is further divided into four sub-neighbourhoods, and the modified likelihood ratio test is performed on each of these sub-neighbourhoods. The procedure is repeated recursively until a decision is reached, or the window becomes so small that a significant decision cannot be made. This method might be promising but the idea of repeated deep quadtree-like partitioning seems to be cumbersome. The method proposed in Abbas and Farooq (2018) applied the Bayesian Rough Decision Tree (BRDT) classifier to improve the accuracy of human skin detection. Quadtree classified vector quantization (QCVQ) method was used in Chen et al. (2014). This method first partitions a quadtree into its usual segmentation and then classified into smooth and high-detail blocks. The authors claim that the scheme yields better retrieval performance compared to the well-known vector quantization (VQ)-based image retrieval methods. However, even though quad-trees are very good on images with large areas of a single colour, which eventually become compact, they have proven to have poor shape analysis and poor performance on pattern recognition due to their inability to compare two images with different translation or rotation efficiently, especially with an image that has different colours for every pixel. Other tree-based structures similar to decision trees or quadtrees include; Smit et al. (2013); ElFkihi et al. (2006, March); Khan et al. (2010, September); Zhang et al. (2012, November).

All these methods are quite different from our proposed system. The decision tree is only as an analytical decision/ visualization support tool and might not be proficient for multidimensional or spatial analysis. The quad-tree as we have mentioned is not balanced and that is not good for efficient computation. Moreover, for an image that has different colours for every pixel, this will involve very tedious partitioning, thereby losing effect. Additionally, the four children constrain, and constant partitioning limits proper utilization of the leaf nodes of the quad tree. Moreover, in neural network (NN) methods, the training stage may take a long time if the number of training patterns is very large (Sun, 2010).

5.6.4 Proposed system (Pmkd-tree):

Our proposed tree structure is an integration of the methods we described in Samson et al. (2018); Samson and Lu (2018); Samson et al. (2017); Samson and Lu (2016). In these materials, various related spatial indexing and modelling mechanisms including a description of improvement strategies for spatial structures and techniques were presented; nevertheless, this work is an aggregate classic that produces an efficient technique for spatial modelling of human skin related pixels. The basic concept/idea behind our technique in this current work, is the fact that skin pixels (especially on the face), are hardly isolated. That is, once a skin pigment is encountered at a certain position/location, there is a high probability that the neighbours are skin. As such, selecting a tiny fraction of the pixels, in that bounded area, guarantees a satisfaction of the skin/no-skin classification condition. Therefore, we build an effective k-dimensional tree structure, for partitioning an input image. After the partitioning, we **interpolate** quickly (using inverse distance weighting - ***idw***) through the leave nodes using only a very few sample pixels (**r** out of $\boldsymbol{\Omega_0}$ ------- $\boldsymbol{\Omega_0}$ **is the total pixels in the leave bounding box**) from the leave node bounding box. Finally, only the leave bounding box where all **r** satisfies the skin/no-skin condition in 5.5.6 (i) are returned.

i. Comparison to common colour thresholds for human skin classification.

Chen et al. (2016) described several colour thresholds for modelling skin colours. However, researchers including Kolkur et al. (2017); Jati and Selvam, (2008); Peer and Solina, (1999); Baskan et al. (2002) and Ali et al. (2018), adopted similar RGB/HS colour models for human skin identification and possible recognition. These models fall within a given threshold for all skin colour types, (Figure 45)

(a) RGB (R > 95 and G > 40 and B > 20 and R > G and R > B and | R - G | > 15 and A > 15)

(b) HS >>0.0 <= H <= 50.0 and 0.23 <= S <= 0.68

Figure 45: (a) and (b) shows common HSL, RGB colour model as adopted in Kolkur et al. (2017); Ali et al. (2018); Jati and Selvam, (2008); Peer and Solina, (1999) and Baskan et al. (2002).

Our study is disputing their premise and assumption, as it falls short of reality for certain human skin colour code. Three main colours (***Red, Pink, Brown***) pose the most problem in human skin colour threshold setting. Of course, this is because they are very close in shades to the red colour underlying the human skin Jablonski (2006). After an exhaustive toning on both the HSL and RGB colour channels, in order to enhance insensitivity to luminance, we came up with a more efficient threshold (see Figure 46).

(a) HS (H>=0.0 && H <=30.0 && S >=0.10 && L1 >=0.10);

RGB (R > 50 && R < 220 && G >40 && G <200 && B > 40 && B <180 && R > G && R > B && |(R - G)| > 20)

Figure 46: (a) and (b) shows common HSL, RGB colour models (our suggestion)

It was however noted that these colour models do not perform efficiently when applied discreetly. Hence, we tested a combination of both and the result is found in Figure(s) 47 and 48 respectively.

Figure 47: RGB + HSL shows the result of the combination of HSL, RGB colour models (other methods)

Figure 48: RGB + HSL shows the result of the combination of HSL, RGB colour models (our suggestion)

Notice the high rate of false hit on the other methods with combined colours in Figure 47. Most parts of the skin area in some images were not captured. In some images, non-skin areas were captured as skin. Our suggestion in Figure 48 shows an improvement to these problems and this improvement contributes to the high precision and accuracy of our spatial model described in the next section. A smoothed version of the third image in Figure(s) 47 and 48 respectively, using our model ***(pmkd-tree)*** is shown in figure 53(h).

ii. Method description (Spatial modelling):

The classification/prediction procedure in section 5.5.6 (i), for predicting skin and non-skin pixels in an input image based on a defined colour threshold, is normally performed (in most state-of-the-art systems) by testing individual pixels (Pixel-by-pixel or pixel-wise operations). However, this process is very slow with high degree of false hits (as shown in table 1), but can actually speed up and perform better if enhanced by a k-dimensional data structure like our proposed model (**P*mkd-tree)***. The model largely depends on an arbitrary value **μ,** which determines its performance. With an exhaustive experiment, **μ=25** was established as the most fitting value for any image type and size, though that depends on the underlying task. For ***dimension k=2,*** and for an image of ***size*** = DataL, given that **Δ** = (DataL)/**μ**) $^{1/k}$, the number of pixels to be processed reduces from DataL to **ð = r * Δ *Δ**, where **r** =5/Ω_0 and Ω_0**,** is the number of data elements on each leaf node of the proposed tree. This means that for an image of ***size 3000 px***, rather than processing the entire pixel elements, only ***605*** pixel elements are processed yet an efficient outcome is achieved. The proposed tree structure is described in section 5.5.6 (iii) --- 5.5.6 (v). However, the basic idea is as follows. In dimensions ***k=two (2)***, the image (x, y) pixel positions and RGB values are extracted and stored in a ***k-dimensional array*** of size DataL. The array is partitioned following the procedure outlined in section 5.5.6 (iii). **Five (5**) significant sample pixels **(r)** are selected from each bounded leaf node of the tree, where each leaf node contains a certain fixed amount of Ω_0 pixels (explained in section 3.2.1). Through interpolation, other pixels are classified based on the value of the value of these **(r)** selected pixels. Finally, only leave bounding boxes, where all **(r)** pixels pass the skin classification test are returned.

iii. ALGORITHM 1:

Partitioning: Building the Pmkd-tree

Given an **ŋ*m** image, where **ŋ**= 50, **m** =60

1. **DataL=array of pixels**=50x60 =**3000px** .....**EQN 1**

   For dimension k =2

2. Let **ex1=1/k**;

3. Let Δ = ceiling (**DataL)/ μ**) ^{ex1} .....**EQN 2**

   ***Note: μ** is an arbitrary number (here, we choose **25**. See the effect of different **μ** value in Figure 51)*

THEN

4. Partition **DataL** into **¥** sections on the first dimension,

   Where ¥= DataL/ Δ .....**EQN 3**

THEN

For each partition **P** in **¥**,

5. Partition **P** into **¥1** sections on the second dimension (for 2 dimensions)

   Where ¥1= ¥/Δ .....**EQN 4**

THEN

6. Return leaf **(**bounding box of array of Ω_0 points in leaf node **(see section);**

The procedures above force the tree to stop at depth k=two (2), thereby storing the data items on the leaf nodes of depth 2.

In algorithm 1, we perform the partitioning in two (2) dimensions. After finding **Δ (the number of partitions in each dimensions)** in **EQN 2**, we divided the **ŋ*ŋ array of** image pixels (where **ŋ**= 50 and **DataL= ŋ*ŋ)**, in the first dimension, i.e. **¥= DataL/ Δ. E**ach P' partition containing an array of size **¥,** is further divided by **Δ in the second dimension**, to get the leaf node of size **¥1= ¥/Δ.** This means that, for *all leaf nodes* (**enclosed in a bounding box**), there is a total pixel of **¥1=Ω_0,** in depth **k (the tree depth is directly proportional to the dimension)**. Now that we have the value of Ω_0, we find **ð** (**the expected significant pixels in an image for classification** – section 5.5.6 (iv)).

iv. ALGORITHM 2:

Finding ð

DataL → Total pixels in an image

ð → Total expected significant pixels in an image for efficient skin/no-skin classification.

CONST Ω_0 (section 5.5.6 (iii) → $\Omega_0 = [¥1]$**)** **...EQN 5**

THEN

1. Select **r sample** pixels (**r** is explained in section 5.5.6 (v)) from each Ω_0
2. Compute **ð = r x Δ x Δ**

   Δ derived from **EQN 2,**

Total expected significant pixels in an image: $ð = r\,x\,\Delta\,x\,\Delta$ **...EQN 6**

Algorithm 2 describes the calculation of **ð** →the total amount of pixels that are **significant** for the skin/no-skin classification purpose**. Note: ð** in **EQN 6** can vary greatly, depending on the size of the image. Nevertheless, the **r** (**EQN 8)** and Ω_0 **(EQN 5)** remain constant for all images and it is not arbitrary, rather, **r** is the sample pixels selected from each leave nodes, while $\Omega_{0\,is}$ the number of pixels on each bounded leave node (typically, $\Omega_0 \cong$ **24 —section** 5.5.6 (vii)) that has to be compared. Also, note that the number of occurrences of **Δ** as in **EQN 6** will directly depend on the number of dimensions of the image. Therefore, for a **k** dimensional space, we would have **$r x \Delta x \Delta \cdots \ldots kth_{\Delta}$.** Note that **EQN 6** *will greatly reduce the size of computation from comparing* ***3000*** *image pixels to comparing only* ***605*** *image pixels in finding pixels that correspond to human skin.* **I.e.** for our sample image of **size DataL=3000, $\Delta = (3000/25)^{1/2} = 11$.** Let **r = 5 (EQN 8)** be the number of sample pixels to be compared out of Ω_0**=24** of each leave node pixels, then from **EQN 6**, **ð =5*11*11=605**. So, rather than comparing all **3000** available pixels we will only compare **605** (see section 5.5.6 (vii))

v. ALGORITHM 3:

Finding r

Sort the array of length Ω_0 **(total pixels in each leaf node)** from **(EQN 5)** in any axis of the dimension

THEN:

1. Find the pixel at the middle **P1** □ Ω_0 [Ω_0 /2]
2. Find the pixel at the top-left-corner **P2** □ Ω_0 [0]
3. Find the pixel at the bottom-right-corner **P3** □ Ω_0 [Ω_0 -1]
4. Find the pixel at the top-right **P4** □ Ω_0 [(Ω_0 /4)]
5. Find the pixel at the bottom-left-corner **P5** □ Ω_0 [(3/4* Ω_0)]
6. **Φ** = [**P1, P2, P3, P4, P5**] ...**EQN 7**

Thus **r** = count (**Φ**) = **5** ...**EQN 8**

We have computed **r** as **r = 5**, in algorithm 3 (**EQN 8**). This means only **r** → **5/Ω_0** pixels from every leave node is to be processed and **r** is constant. With this enhancement, we have reduced the computation from **3000** ***to*** **605** (see **EQN 6**).

vi. Pmkd-tree (The main proposed structured):

In an attempt at data reduction for time effective computation, our algorithm starts by compressing image quality and reducing the size of the image, thereby achieving only 60% of the image size and quality. This pre-processing technique increases the speed of the application. Similar to the KD-tree *Bentley (1975) and X-tree (*Berchtold et al. (2001*) in terms of logic*, the ***Pmkd-tree*** (typical structure figure 6, output figure 8) performs a partitioning (albeit preserved) of a k-dimensional tree (2d in our case); Building a static ***Pmkd-tree*** from *n* points has the average time complexity of O(log n), the same with skin detection operation. The ***Pmkd-tree*** partitioning described in section 5.5.6 (iii) is preserved in the sense that the tree does not go deeper than a maximum depth of 2, For dimension=2, depth of 3 for dimension = 3 etc. After partitioning in the first dimension, all **Δ** partitions are placed in an array, which forms a forest of internal nodes on the root (see Figure 49), where each member is a sub (***Pmkd***) tree. Each member partition is further divided into **Δ'** partitions (section 5.5.6 (iii)) ***recursively***. Similarly, all **Δ'** partitions **(EQN 4)** are also placed in an array, which also forms a forest of **leaf nodes** (with **Ω_0 pixels**) on each leaf nodes. All **Ω_0** pixels **(EQN 5)** on each leaf node is stored in a bounding box. In Figure 49, a typical ***Pmkd-tree***, showing the root node as a forest of internal subtrees, internal node as a forest of leaf bounding

boxes and leaf nodes containing the image pixel data. Note that if the size of the pixel array **DataL** is not even, the last leaf node partition will be extended to a **super-node**. The idea of a super-node does not affect the performance of the tree, because in terms of pixel classification, the mid pixel in the bounding box carries the most classification weight (see section 5.5.6 (vii))). For disk based structures, the algorithm is **packed**, in the sense that a total $\boldsymbol{\Omega_0}$ of pixels are recursive stored on each leave, where all leave nodes on the same partition are further packed into an array and stored on the upper (internal) node, until the **root** is reached. The output of the tree is shown in Figure 51 with different values of **μ**. The **μ**, **(**section 5.5.6 (iii)) is the determinant of tree behaviour and performance. For this work, **μ** =25 (Figure 51 (e)) has been determined as the best value (after an exhaustive experiment) for efficient performance for an image of any size.

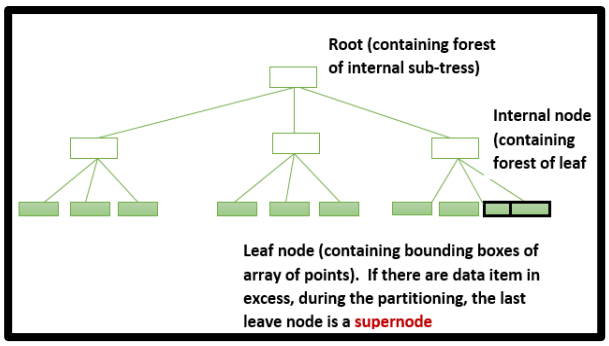


Figure 49: Typical Pmkd-tree in two (2) dimension

vii. Interpolation Process:

For pixel matching purpose, following the partitioning in section 5.5.6 (iii), the matching procedure is expected to process:

$\boldsymbol{\Omega_0} * \Delta * \Delta \subseteq DataL$ individual pixels of the image **(A)**

However, this will make the classification process highly inefficient and impractical. Therefore, we compute **r** (section 5.5.6 (v)), to reduce the computation to $r * \Delta * \Delta \subseteq$ ð. Afterwards, the interpolation procedure below is followed to classify the pixels into skin or non-skin, using only r **sample pixels**.

Considering the pixels **(P)** in **DataL, assume P_i, i=1…. DataL** to be a set of **S** spatial event {$\mathbf{p_1}$, $\mathbf{p_2}$ **…**$\mathbf{p_S}$}, let **t** be some colour threshold (section 5.5.6 (i)) and Figure 46) for classifying $\mathbf{P_i}$ as skin/no-skin i.e.: $t\ (P)\ => T\ ||\ t\ (P)\ => F;\ \forall P_{\ i=1....L}$ **(B)**

Proposition: If **t** in **(B)** is accurate, only **ð** (defined in section 5.5.6 (iv)) pixels instead of **DataL** needs to be processed and is significant for the prediction in **(B)** to hold**, with an** accurate, precise and fast **solution**.

Proof 1:

Let instances *I* {**P**} in **(B)**, be all **P** ∈ n, n is a vector <instance-id, spatial event type (**p**), location>, where locations belong to a spatial framework F (likened to our ***Pmkd-tree structure***),

Then,

∀ P∈S in **(B)**, since there is a discrete hybrid partition P'of *I* based on a spatial proximity (***SP***), and a close ***SP*** (***SP'***) over *I*, then, a spatial inferential rule is discovered faster and more efficiently.

Following **Proof 1**, we can now say that, for any subset $p'_i \subseteq$ P'**,** where i=1**…** $\Delta * \Delta$ (as seen in tiny boxes in Figure 51(e)), in dimension ***k***, assuming total pixels is **DataL**, and **μ** =25,

Then sum of pixels in p'_i is a ≅ $\mathcal{C} = \boldsymbol{\Omega_0}$ in **(A).** **(C)**

$\lceil (DataL\ /\mu))^{1/k} \rceil \cong \mathcal{C}''$ **(D)**

$\Longrightarrow \lfloor (DataL/\ \mathcal{C}'')/\mathcal{C}'' \rfloor = \mathcal{C}$ **(E)**

$\Longrightarrow \mathcal{C}$ in **(E)** is ≅ $24\ \ for$ all image sizes. **(E1)**

For example, let $DataL = 3000, and\ k = 2$

Hence $ceil\ \left(\frac{3000}{25}\right)^{1/2} = 11$

Thus from **(D)** => *floor* (3000/11)/11≅24

∴ = $\boldsymbol{\Omega_0}$ 24 **(F)**

Proof 2:

Spatial autocorrelation measures the similarity between ***samples*** for a given variable as a function of ***spatial distance*** (Rossi and Queneherve, (1998); Legendre, (1993); Samson et al. (2014)). **Thus, following the above proposition with the** assumption that spatially distributed objects are highly spatially correlated; we define q (the ***spatial distance SP***) as any spatial construct:

q = {q : q ∈ ***SP***}, Then $q = \{0,\ if\ SP'(\ p'_i\ ), \geq 1,\ otherwise$ **(G)**

$\therefore \forall\ p'_i$ **in Proof 1,** since spatial autocorrelation occurs due to correlation of a variable with itself through space (Chen et al., 2011), we assume q = 0, in **(G)**. That is to say, individual observations made from the chosen samples include information present in order observations in all $p'_i \subseteq \boldsymbol{P'}$, so that the actual sample size, **r**, is less than the number of observations, **DataL**.

Now, since $\boldsymbol{\Omega_0}$ **(**leave node of the partition in **(A))** is a bounding box containing $\mathcal{C}$ **(**expatiate **in (F)) highly** correlated **pixels, it the means that (following the proposition in (G))**, there is a high probability that if the midpoint pixel ($r\mathcal{C}mp$) in the bounding box, passes the threshold test:
t ($r\mathcal{C}mp$**):** $r\mathcal{C}mp \rightarrow$**T** in **(B).** Then for all $\boldsymbol{\Omega_0}$ pixels, **t (**$\boldsymbol{\Omega_0}$**) => T** **(H)**

we select the four corner points (TopLeft ($r\mathcal{C}tlp$), TopRight ($r\mathcal{C}trp$), BottomLeft ($r\mathcal{C}blp$), BottomRight ($r\mathcal{C}blp$)) from the boundary of the bounding box plus $r\mathcal{C}mp$, as the five significant sample points for **interpolation (see Figure 50. As** such, **r** is the length of the array $[r\mathcal{C}mp, r\mathcal{C}tlp, r\mathcal{C}trp, r\mathcal{C}brp, r\mathcal{C}blp] \rightarrow$ **five (5).** **…(H1)**

Prediction by interpolation Using Inverse distance weighting (*idw*):

Interpolation (Figure 50) is a way of predicting values for cells from a limited number of sample data points. The diagram in Figure 50, depicts the prediction procedure as explained in **(H),** the white points in **(c)** and **(d),** are arbitrary unknown pixel values**.** The blue point at the middle ($r\mathcal{C}mp$) is a known sample point, which carries a greatest weight **(**$\boldsymbol{w} = d(r,\ r_j)^2$**)** —see **(J),** where ***d*** **=distance.**

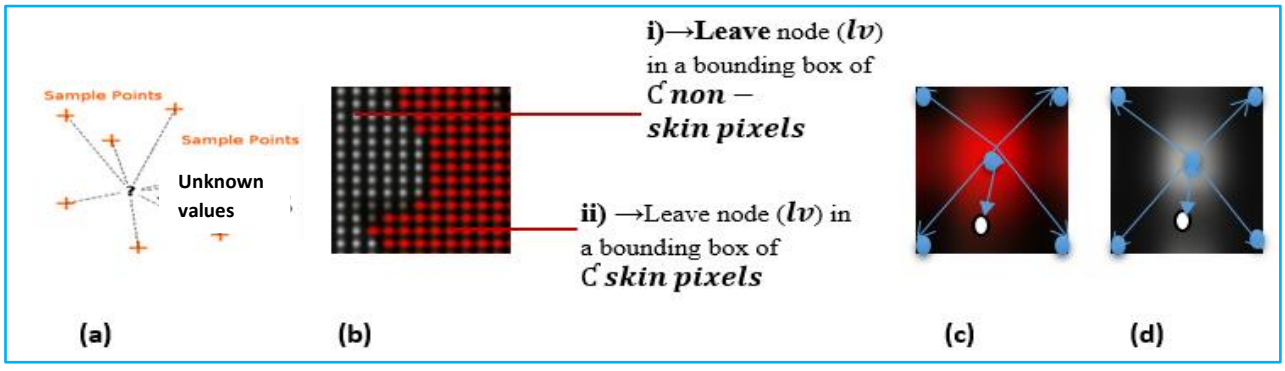


Figure 50: Interpolation procedure
(a) underlying idea of predicting new unknown cell values from known ones based on known sample points, (b) Expanded cells showing skin and non-skin pixel (in their bounding boxes) as classified by

Pmkd-tree cropped from figure 10f, (c) Expanded single cell from skin area from figure 7(b-ii), (d) Expanded single cell from non-skin area from figure 7(b-i).

Assumptions:

Our interpolation method is based on the two assumptions below.

1. Based on P′ in **Proof 1,** then it will be accurate to say, q=0 in **(G). That means the** distance between $r\mathcal{C}mp$ and all the **r** in **(H1)** = 0.
2. For every cell like **(c) and (d) in figure 7(b),** if $(r\mathcal{C}mp)$ passes the skinness **test in (B),** then there is a high **probability** that the boundary blue points will pass the test as long as 1 remains true

Consequently, we set a search radius of size $\mathcal{C}see\ (F)$. However, only **r** as in (**Proof 2 - (H1))** of these points are used for testing. From these selected known sample points and their threshold values, we have a 1d list of tuples. $\mathcal{Z}$= **[(r_1, t_1), (r2, t2)… ($r_{p'}$, $t_{p'}$)]** **(H2)**

∴we find a discrete assignment of the *unknown* **r** function on each partition $p'_i \subseteq \boldsymbol{P'}$ using inverse distance weighting (***idw***) with a power of 2:

i.e. $t(r)$**: r** →**T,** **r**∈ P′ ⊆ P **(I)**

$$\Rightarrow \quad t(r) = \{\frac{\sum_{j=1}^{5} \blacksquare \left(\frac{1}{d(r,\ r_j)^2}\right) t_i}{\sum_{j=1}^{5} \blacksquare \left(\frac{1}{d(r,\ r_j)^2}\right)},\ if\ d(r,r_j) \neq 0\ for\ all\ i \quad t_i\ , \quad if\ d(r,\ r_j) =$$

$0\ for\ some\ i$ **(J)**

Following the definition in **(J), the function** $(t)(r_{j,\ i=1\ldots..\mathcal{C}})$ = t_i **holds,** because, since q=0 in **(G)**, then it follows that $d(r,\ r_j)$ **in (J) = 0,** for all the pixels ($\mathcal{C}$) in the leave node bounding box**.** Therefore, we are left with the task of finding and returning all cells (**[$\mathcal{Z}_i$]**) as in **(K)** that is **True**

[$\mathcal{Z}_i$], i=1….5list of all ($\boldsymbol{P'}$**), that pass the threshold test** $\mathcal{Z}$**→T** **(K)**

Note that combining the interpolation processes with our fast, spatial search structure **P*mkd-tree*** achieves an efficient log *N* interpolation method performance, which is highly suitable for large-scale problems.

So, if we accept the assertion in **(J), and (K)** is true**,** then the calculation in **(A) reduces to:**

$\boldsymbol{\Omega_0} * \Delta * \Delta \leq DataL \ \Rightarrow \ r * \Delta * \Delta \leq$ ð … **(L)**

Summary: From the example in **(E)**, $C'' = 11$ and from **(H1), r=five (5),** thus:

Total expected significant pixels in an image for efficient skin/no-skin classification **(ð)**

→ **ð = 5*11*11= 605 < (Ω_0=24) *11*11 ≅ 3000**

i.e. **ð = 5*Δ^2** **(M)**

If we convert the calculation in **(M)** into **percentage i.e. find** the ratio of the number of operations to the size of the input in percentage and multiply by 100 → **(605/3000) * 100,** the answer will be approximately **20. Thus,** we conclude that:

The percentage of pixel needed to detect human skin from any image → $\frac{ð}{DataL} * 100$ **≅20**

∴Only 20% of the pixels in an image is required to classify the pixels, in a skin detection procedure

This model is true for all mage types and sizes, as long as **t in (B)** is accurate**.**

Finding the human skin pixel:

Now that we have described our interpolation procedure, using ***idw***, let us look at how the ***Pmkd-tree*** carries out the classification process. We used two (2) colour models HSL (see Figure 44a) and the RGB colour models (see Figure 44a), in identifying human skin presence. Initially, the program starts by extracting all the pixels from the image and then stores them in a k-dimensional array. Then the RGB values of each pixel in the array are converted to HSL values. The array is then partitioned following the procedure in section 5.5.6 (iii). After the partitioning, a forest of sub-trees containing all the subtrees, are then stored on all internal nodes. Figure 51 shows the output of the tree with different values of **μ**. Figure 51 (a) is the actual test image, Figure 51 (e) with **μ** = 25 (see section 5.5.6 (iii), for how we chose **μ**), shows a better performance.

(a) **(b) => *μ* =20** **(c) => *μ* =40** **(d) => *μ* =100** **(e) => *μ* =25**

Figure 51: (a) the original image (b, c and d) outcome of the partition based on different values of μ.

From the leaves of each of these sub (internal) trees (the tiny rectangles in Figure 51 (e)), containing $\mathcal{C}$ pixels (see **(F)** for meaning of $\mathcal{C}$), only **r** pixels (as explained in **H1 of proof 2)** are selected. An array is created of the **r** pixels (as in equation 7) and the array is stored on a higher array on the internal node, producing a **1d** array ($\mathcal{Z}$) as in **(H2). The** elements in $\mathcal{Z}$ are compared **recursively** to match the skin/no-skins threshold following the procedure in section 5.5.6 (ix) and only $i \in [\mathcal{Z}_i] \subseteq \mathcal{Z}$ **elements** as in' **(K) are returned.**

viii. ALGORITHM 4: Detecting skin pixels

Consider **r** in **EQN 7 (**an array of five (5) selected candidate pixels for skin classification selected from each leaf node)

Then find the skin pigments as follows:

Let β = colour threshold for skin (defined in section 5.5.6 (i), Fig. 48)

Let Spx= qualified **r**

Let β0= an empty list to contain skin pixels (i.e. S**px**)

THEN

∀ Φ

$\forall_j \in \Phi$ j=1…… r

a) Extract RGB coordinates **of j**

i. *If* $(j_{RGB}) = \beta_{RGB}$

b) compute the HSL **values of j**

i. *If* $(j_{HSL}) = \beta_{HSL}$

c) add **px** to **β0**

Return β0 (2D array of skin pixels**)**

5.6.5 Result/discussion:

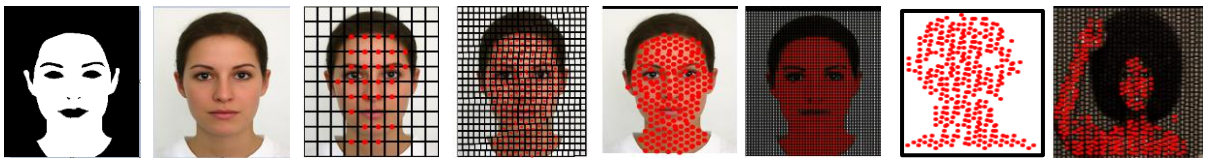
Our models propose an improved colour threshold (section 5.5.6 (i), Fig. 48) and multidimensional spatial (tree) structure (section 5.5.6 (i)), for effectively detecting human skin from an image of any form. Below we have presented the result of the model. The results show that the structure is very versatile as it is promising, showing tendencies of greater prospects (as shown in table 7). For instance, by performing a little bit of **geometry** on the reverse aspect of the model (as in Figure 52), some facial features including the face nose, eyes mouth and so on, can be detected. In Figure

52, the non-skin area has been marked with red points by ***Pmkd-tree.*** Using some simple distance metrics, the head and neck could be extracted. Additionally, working out the position of facial features could help find the nose, mouth and eyes.

(a) (b)

Figure 52: (a) is the reverse aspect of our test image, non-skin areas are shaded (b) cropped portion of the image isolating facial features

Figure 53, shows the various stages of the procedures of the ***Pmkd-tree***. The ground truth image Figure 53 (a), original image Figure 53 (b) and result of various stages of the skin detecting process. Figure 53 (f) is the result (with **μ** = 25). The red points on the face are midpoints of the bounding boxes on each leaf node where equation **(K) holds.** Figure 53 (g) shows just points that represent skin area. It is apparent that **μ** affects the performance and behaviour of the tree. In Figure 53 (c), where **μ** = 500, some part of skin pigments was not detected, this will definitely give rise to a high rate of false hits. Even though Figure 53 (d) --where **μ** = 100—looks promising, there is still a tendency of some measure of false negative hit. Figure 53 (e) shows the outcome of Figure 53 (d) without displaying the tree. At **μ** = 25 in Figure 53 (f), a perfect result was achieved. Figure 53 (h) shows how the ***pmkd-tree*** was used to smooth out the result of the image in section 5.5.6 (i)), Figure 45.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 53: (a) ground truth image, (b) the original image (c) output when μ = 500 (d) output when μ = 100 (e) midpoints of leave bounding (where (K) holds) shown in red. (f) The expected and final result of the skin detection process extracted skin areas marked as red points. (g) Points extracted from an image using pmkd-tree. (h) pmkd-tree smoothed image of the image in section 3.1 figure 5

In Figure 54, we show the result of applying the algorithm in finding skin of varying skin types, complexion, illumination, shade, pose, position etc. The images in Figure 55 are skin pigments, detected using the same technique, however, the tree boundaries are not displayed.

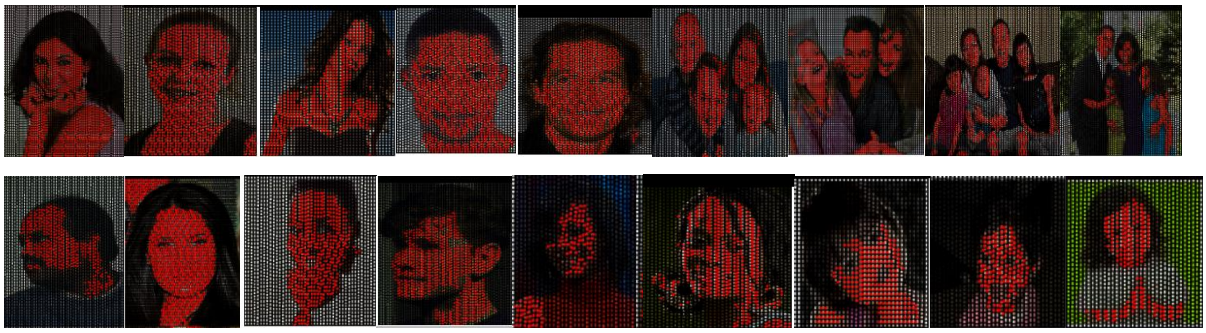


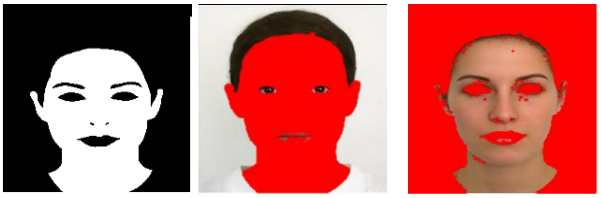
Figure 54: other images, showing detected skin using our proposed model.

Figure 55: more images tested (tree not displayed)

5.6.6 Evaluation

In order to evaluate the performance of the tree structure against the commonly used methods (*Pixel-by-pixel / pixel-wise operations* and quad-tree like enhancement structures) adopted by many authors. We have compared the tree performance with these techniques (Figure 56). The image in Figure 56, shows the performance of the various models. In Figure 56 (a), we have the ground truth image, Figure 56 (a-i) shows the skin area and Figure 56 (a-ii) highlights non-skin areas as identified using the *pixel-wise* technique. No enhancement was applied; thus, each pixel was checked individually based on our purported colour threshold for skin pixel classification in 5.5.6 (i), Figure 45. In the first picture in Figure 56 (b), we show the same image with the skin area mapped out with the ***Pmkd-tree***. In Figure 56 (b-i) the tree boundary is not displayed and in Figure 56 (b-ii), the reverse effect of the tree was depicted, showing non-skin areas as identified by the tree. Figure 56 (c) is the result of applying the improved packed quad-tree structure` (***Pquad-Tree****) to the skin prediction procedure based on our improved colour threshold in* 5.5.6 (i), Figure 45 skin area mapped out with the ***Pquad-Tree*** *i*s shown in Figure 56 (c-i) without displaying the boundaries. In Figure 56 (c-ii), the reverse effect of the ***Pquad-Tree*** was depicted, showing non skin areas as identified by the tree.

Pixel-by-pixel

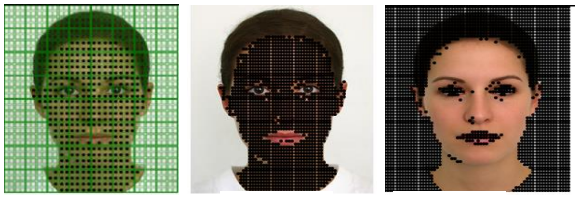


(a) (a-i) (a-ii)

Pmkd-tree

(b) (b-i) (b-ii)

Pquad-Tree



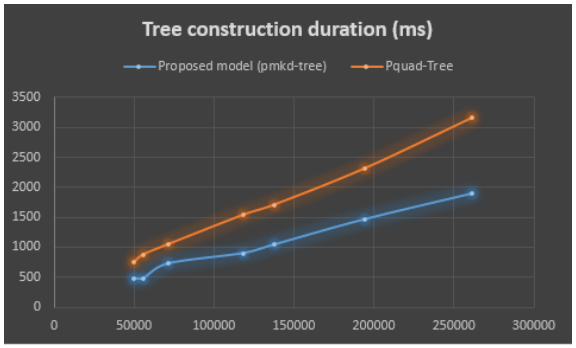
(c-i) (c-ii)

(c)

Figure 56: comparison of performance and elapsed time between the three method of study
(a) pixel-by-pixel skin identification procedure, (a-i) skin pixels (in red) extracted using the method, (a-ii) reverse process, showing non skin pixels in red, (b) result and performance measure of Pmkd-tree (proposed model) skin identification procedure, (b-i) extracted skin area in red (red points represents the midpoint of the leaf node in the region), (b-ii) reverse process showing non- skin area and (c) result and performance measure of the Pquad-Tree skin identification procedure, (c-i)) extracted skin area in black points, (c-ii) reverse process showing non- skin area.

i. Performance graph

In Figure 57(a), a comparison of time of construction between the ***Pmkd-tree*** and the ***Pquad-tree*** is shown. Needless to say, that the quad-tree deep quad partitioning strategies has a negative effect on the speed performance of the structure. Though the quad structure performs fairly in terms of classification (Figure 56), a little improvement might be necessary in order to speed it up. As can be seen, the worst performance in terms of time consumption for pixel classification is the ***pixel-by-pixel* technique**, followed by the ***Pquad-tree*** method (Figure 57(b)). This means that despite the fact that there are no structures to build in the ***pixel-by-pixel* method**, the method cannot improve beyond a time complexity of O(n). Note that the timing here includes the pixel extraction time, array manipulation, tree partitioning and pixel classification.



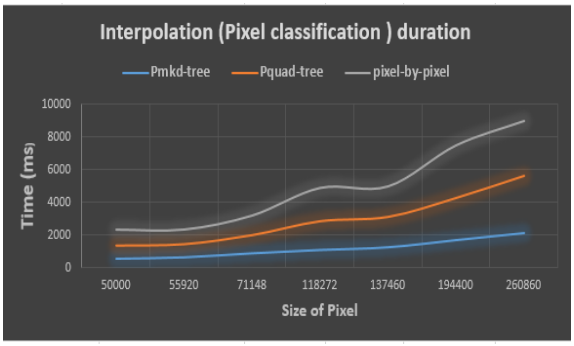


Figure 57: Time of construction and classification

(a) comparison of times elapsed between proposed system pmkd-tree and other (b) comparison between size of data and duration of pixel classification between the two methods of study.

ii. Precision and Accuracy

In order to calculate the accuracy and precision of our proposed model, we have prepared a table (see table 7). The table shows the accuracy and precision rate of these three methods in terms of pixel classification. The accuracy and precision calculations were based on the formula in developers (2019), for proper evaluation of our skin/no-skin pixel classification model. The high precision and accuracy rate achieved by ***Pmkd-tree*** is due to the large ***idw*** weight value attached to the midpoint pixel $r\mathcal{C}mp$ of the leave bounding boxes such that $r\mathcal{C}mp$ must be true for any other unknown pixel in that bounding box to be true (as in the equation in **(H)**). Again, the bounding boxes are very tiny (as seen in Figure 50) as such, the truthy/falsy values of the remaining sample pixels in **(H1)** are easy to predict.

5.6.7 Conclusion and future works:

Pixel-by-pixel operation does not have any significant advantage for human skin detection or skin pixel classification, as it is characterised by high false hits and huge time consumption. In this paper, we have presented an improved threshold-based algorithm for recognizing human skin pixels in an image using the combination of RGB-HS colour models. In the quest to speed up the process of skin/no-skin classification, we have proposed and implemented a ***k-dimensional structure***, which not only speeds up the process, but also improves performance. The algorithm can process images of different light conditions including brightness etc. Our proposed model shows very high promising results in terms of precision and accuracy as compared to most state-of-the-art systems. Images from different sources were tested, and the model scaled high. We have also shown that with little geometry, the algorithm can detect face, hand and other features and gestures. From the results presented in table 1, the overall performance of the algorithm, it can be

seen that the proposed model provides a very significant reduction in false detection rates as compared to ***pixel-by-pixel*** testing mechanism and quad-tree like techniques applied in many systems. Quad-trees have been in use for speed up of the detection process, however, we proposed and implemented an improved packed quad-tree, to which we compared our main model (**P*mkd-tree***). Although there is significant improvement as compared to the ***pixel-by-pixel*** techniques in terms of speed and accuracy, the quad-tree structure showed certain drawbacks in terms speed of construction and speed of classification of pixels and this, we attributed to the structures' partitioning strategy. O*ur main procedure performed far better in terms of precision and accuracy* as compared to these other techniques**.** We can boldly say that the proposed approach yields better detection performance measures as compared to that of the state-of-the-art ***pixel-by-pixel*** and quad-tree based techniques with a significant reduction in time and computational cost.

For future work, we will apply the proposed system to further biometric feature detection and gestures. We shall also look into parallelizing the structure in order to further improve speed.

Table Measurement of Accuracy and Precision of proposed structure

Image one (1)

Method	Total No of pixel in image	Skin pixel detected	Time (ms)	Non - Skin pixel present	True Positive	False Positive	True Negative	False Negative	Precision / Accuracy (%)
Pixel-by-Pixel	**50,000**	20,510	1173	29,490	18,926	1584	29,490	0	**92.37 / 96.83**
Pquad-Tree	**50,000**	18,000	803	27,468	18,000	0	27,648	926	**100 / 98.01**
Pmkd-tree	**50,000**	18,720	473	32,064	18,720	0	31,074	206	**100 / 99.59**
Ground Truth	**50,000**	18,926	648	31,074	-	-	-	-	-

Image two (2)

Method	Total No of pixel in image	Skin pixel detected	Time (ms)	Non - Skin pixel present	True Positive	False Positive	True Negative	False Negative	Precision / Accuracy (%)
Pixel-by-Pixel	**137460**	15,866	1,914	121,594	14,482	1,384	121,594	0	**91.26 / 98.83**
Pquad-Tree	**137460**	7,368	1,597	90,888	7,368	0	90,888	7114	**100 / 71.49**
Pmkd-tree	**137460**	11,920	1,309	127,440	11,920	0	122,978	2562	**100 / 98.13**
Ground Truth	**137460**	14,482	1,060	122,978	-	-	-	-	-

Image three (3)

Method	Total No of pixel in image	Skin pixel detected	Time (ms)	Non - Skin pixel present	True Positive	False Positive	True Negative	False Negative	Precision / Accuracy (%)
Pixel-by-Pixel	**260820**	19,124	3,602	241,696	18,449	675	241,696	0	**96.47 / 99.74**
Pquad-Tree	**260820**	6,188	3,555	214,758	6,188	0	214,758	12,261	**100 / 94.74**
Pmkd-tree	**260820**	9,744	2,847	252,312	9,744	0	242371	8,705	**100 / 96.80**
Ground Truth	**260820**	18,449	1,908	242371	-	-	-	-	-

Image four(4)

Method	Total No of pixel in image	Skin pixel detected	Time (ms)	Non - Skin pixel present	True Positive	False Positive	True Negative	False Negative	Precision / Accuracy (%)
Pixel-by-Pixel	**71148**	24239	1346	46909	19,666	4573	46909	0	**81.80 / 93.57**
Pquad-Tree	**71148**	13,584	965	35,568	13,584	0	35,568	6082	**100 / 88.82**
Pmkd-tree	**71148**	20,040	645	51,264	19,666	374	51,264	0	**98.19 / 99.47**
Ground Truth	71148	19,666	581	51,482	-	-	-	-	-

5.7 Fast novel clustering algorithm for human face detection

A. Description:

Humanly related tasks like face biometrics, hand detection/tracking, face detection, personal identity, and facial expression analysis, are applications that fall within the fields of image analysis, computer vision or image processing. Face detection/recognition especially, has proven useful in a variety of domains such as healthcare system and biometric system. A physical aspect of the human face is sometimes used to work out their age, gender or basic facial expressions. Many advanced novel methods are already proposed to improve the performance of existing models for face detection/recognition. However, recent research shows that a number of challenges, including missing data (pixel), high level of noise, blur and brightness still presents countless detrimental effects on these models. Observation suggests that models capable of recognition from partially observed data, actual input information and blurred or low-resolution images are needed and should be the focus of future research efforts. Based on this, we are proposing a non-biometric, improved RGB-HSL colour threshold, flexible multi-dimensional spatial structure, packed maintained k-dimensional tree *(Pmkd-tree*) for human skin detection, which speeds up clustering procedures for face detection. Our method does not make any restrictive assumptions based on the biometric data, thus, can be applicable to multiple biometrics. Even though the systems do not have to go through all the complicated and tedious computations typical of recent models for the same purpose, the overall results of our skin colour detection program appear quite very encouraging.

B. Method

We have reviewed existing techniques for face detection and highlighted their shortcomings. In this section, we are going to describe our proposed method. Our work is built on the techniques described in Samson et al *(2018) and* the method described in section 5.5. It is a non-biometric procedure for detecting faces from a 2D coloured image. The procedure initially uses an improved threshold of the ***HSV and RGB*** colour channels for human skin colour pigmentation, and then proceeds using an improved clustering technique to find the face. We used images from the ColourFERET dataset NIST (2019), Pratheepan human skin dataset Tan et al, (IEEE T-II, 2012) and various other image of diverse complexion, pose, orientation, age, variation of illumination and sex, selected from the internet. Our method does not make any restrictive assumptions based on the biometric data, thus, can be applicable to multiple biometrics. The threshold for the colour models in Kolkur et al. (2017) and Ali et al. (2018) is not vast, as such not robust for diverse skin

complexion (especially dark skins). Therefore, we have presented an improved model here (see table 8).

Table 8: Comparison between our improved colour threshold and others.

Others	Our model
HSV□ >> ***0.0 <= H <= 50.0 and 0.23 <= S <= 0.68 and 0.0.***	***HSV***□ (H>=0.0 && H <=30.0 && S >=0.10 && L1 >=0.10)
RGB□ ***(R > 95 and G > 40 and B > 20 and R > G and R > B and | R - G | > 15 and A > 15).***	**RGB:** **(**R > 50 && R < 220 && G >40 && G <200 && B > 40 && B <180 && R > G && R > B && |(R - G)| > 20)

In addition to the robust colour threshold, our proposed system, rather than perform a pixel-by-pixel operation for colour segmentation, as applicable in many of the existing systems, relies on a spatial structure: *Pmkd-tree* (maintained n-dimensional tree) for the operation. The algorithm (*Pmkd-tree described in section 5.5.6*) largely depends on an arbitrary value **μ,** which determines its performance. With an exhaustive experiment, **μ=25** was established as the most fitting value for any image type and size, though that depends on the underlying task in two (2) ***dimensions.***

C. Description of system:

After the skin-finding algorithm in section 5.5.6 (ix) algorithm has been completed, then we run algorithm 5 for clustering skin pixels that fall only on the face area, using the geometric calculation in algorithm 5.

Algorithm 5: (Clustering – for Face Finding)

For a single human face in an image

Let ***list*** → an array of all pixels derived from algorithm in section 5.5.6 (ix)

Let ***d*** → a given distance threshold, 150 in this case

Let XLEFT→ value of x at left

Let XRIGHT→ value of x at right

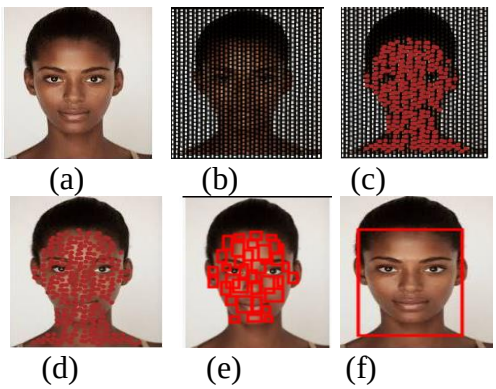
1. Let YMIN → the minimum value of second coordinate (y) in ***list***
   Then
2. ∀ **o** (other points **o** in **list**)
   If distance (YMIN**, o) < d && value of o in** second dimension (for 2d) <= **150** && XMIN > 90 && XMAX < 90
   Then
3. **Add o** to neighbours (**n**) **of** YMIN
4. **Build** a bounding box (***bb***) around **n**
5. **Remove o** and YMIN from ***list***
6. If ***list*** empty
   Stop Go to step 8
   Else
7. **Go to 1**
8. **Return *bb***

D. Result and Discussions:

Result

In this section, we present the results of the experiments, carried out to evaluate the effectiveness of the proposed system in images of varying pose, complexion, size, illumination, expression, position, orientation and pixel quality (see Figure 58).

Based on a novel algorithm Simulated in JavaScript and html for visualization



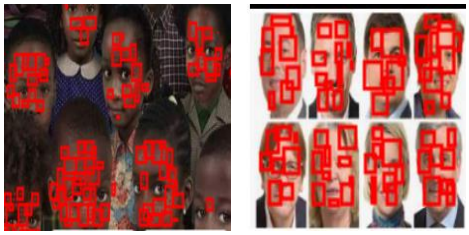
(a) (b) (c)

(d) (e) (f)

Figure 58: (a) original image, (b) *Pmkd-tree* partitioning, (c) skin colour segmentation with *Pmkd-tree*, (d) skin segment of the image, (e) clustered face area with bounding boxes before the merger, (f) detected skin in bounding box.

Figure 59: The images in this section are from various sources, with a bounding box built by our algorithm over their face.

The algorithm is fast and robust, performs optimally, without going through the complicated computation common in state-of-the-art systems. It also works on an image with more than one person (face) as seen in Figure 60. With a little more effort, the algorithm is also expected to perform well for face identification and recognition.



(a) (b)

Figure 60: (a) and (b) shows the performance of the algorithm group images before final merger of bounding boxes.

Discussions:

The clustering algorithm in section 5.6 for finding a single face is distance based. That is, depending on the minimum value of **y** in two (2) dimensions, the clustering starts; gathers all points within a given threshold of **y,** and builds a bounding box around the neighbours of **y**. If there are more points, left in the array, then the procedure starts all over again. This is a brute force search, but suffices for now, however will improve in later versions.

The algorithm is ***devoid of the minp in DBSCAN,*** as all the pixels in the array are equally human skin. Therefore, only a geometric manipulation can determine candidate pixels for the human face based on the position.

A k-dimensional maintained tree structure ***(k= 2 for our example)***, which depth is directly proportional to the underlying image dimension when projected on a 2 or 3d coordinate space, was

also implemented. Even though the system does not have to go through all the complicated and tedious computations typical of recent models for same purpose, the overall results of our skin colour detection appears quite very encouraging. As can be seen in Figure 59, the size of image, age of person orientation of the face, and other distortion do not have a significant impact on the performance of the system. An important characteristic of the system is that, the training depends on a set mathematical/statistical model, as such does not require the extensive training and retraining of sample training set pixels, as typical in many recent systems. Our approach works satisfactorily for now, and has been tested for a general image where it successfully performs face detection. Nevertheless, we need to add more features with sophisticated algorithm to make the system fit for use, for more general and advanced applications.

E. Conclusion:

The human face is a prominent feature of the human body as such plays a significant role in the recognition of a person. Various techniques have been proposed, designed and adopted for face related applications. In this paper, we present an improved spatial model, for face detection. The proposed system is insensitive to image distortions such as pose, expression, position and orientation, skin colour, presence of glasses facial hair, differences in camera gain, lighting conditions, and image resolution. The proposed system is promising and robust. The next phase of the system is to recognize a face and gender; and the intention is to work with a clustered bounding box on the underlying face. The system is non-biometric, as such reserves, a strong potency against security threat like impersonation

5.8 Large Spatial Database Indexing with aX-tree

A. Description:

Detailed in ***Samson et al. 2018***, the project described in this section discusses our main work in this research, geared towards an improved structure for indexing large and high dimensional datasets using spatial data models. The project produced the article titled “Large Spatial Database Indexing with **aX-tree**”. The paper looked into a detailed outline of the main **aX-tree** packing mechanisms for static SDs that accomplishes better space management through an effective packing algorithm. Every other algorithm in this work is based on this particular project. That is to say that this project is the “core” of this research work. This improved structure produces two main advantages: (1) Reduction in the space overhead of the index structure. (2) Production of a more desirable response time (the **aX-tree** index structure has a higher fan-out---i.e. size of leaf node, so the tree will always end up shorter).

B. Major Contributions

The aX-tree proposed model is a novel packing technique for X-tree for stationary SDs. The work presented here greatly contrasts all the models detailed earlier on and any other work in two (2) key perspectives. Firstly, we lay an assertion that this is a rudimentary version of an adjusted X-tree that depends on subordinate sorting is proposed for indexing multidimensional data. Related sort-based models are likely being applied in order spatiotemporal index structures like R-tree but not on the X-tree. Most other methods for handling dimensionality focus strictly on attribute-based vector mapping, dimension reduction (DR), feature embedding, space reduction, point transformation and so on. Additionally, earlier works depend mainly on refining the R-tree and only a small number of modifications are put forward for improving the X-tree (having a grander capability as regards the management of high dimensional spatiotemporal data as research has shown**).** Secondly, our system **aX-tree**, does not adopt a related insertion algorithm as the prevailing X-tree. We distinguish our structure, by initially carrying out a subordinate sort prior to packing the tree nodes. The proposed structure is envisioned to complete the obligation of a filtering, which aim is to decrease the expensive straightforward investigation of geometric objects prompted by the amplified overlap between the rectangles (MBRs). Therefore, an efficient arrangement of the intersecting hyper-rectangles, as well as decreasing the extents of the rectangles (MBRs) will definitely support the efficacy of query, since only a smaller amount MBRs are likely to intersect. This is the main objective the proposed **aX-tree** is set to achieve.

C. Motivation for aX-tree

Lodi et al. (2002) recorded that recent research on system modelling inclines to focus more on sampling and modelling techniques, while neglecting the study and the advantages and qualities of the principal expensive functions. These often leave two (2) main complications

a) The issue of cost in the management of high-dimensional data, and

b) The issue of un-tackled computational complexity.

For example, it is well established and accepted that in small dimensions the utmost efficiency of the organization of the existing X-tree internal node is a hierarchical organization that **matches the tree height to the quantity of necessary page accesses. Likewise,** it is established that for very high dimensions, a linear organization of the directory nodes is more efficient. Therefore, we can state that the X-tree is an ineffective structure for managing high dimensional data. The reason for the above statement is, while it is quicker to carry out operations on the linear part of the tree (i.e. visiting lesser number of tree paths), there is still a near high implementation cost emanating from the overhead created by the super-node (in cases where the query does not affect the entire MBR of the super-node). Additionally, the structure sustains the cost emanating from **overhead**

created **by traversing the nodes of the hierarchical part of the tree**. **Furthermore, in several extreme cases, the X-tree structure linearizes thereby, causing inefficiency in memory management.** An observation is also been made that in higher-dimensional data, numerous geometric data structures (X+-tree, which grows to further splitting of the super-node, thereby degenerating to a clipped or disjoint region Doja et al. (2012), X- tree – Berchtold et al. (1996)) fails to work capably. Most of the current variants of the X-tree presented to enhance the X-tree are either grid or cell based. Grid or cell -based indexing methodology requires point transformations for storing S, and therefore fails to deliver good spatial clustering. The drawbacks and shortcoming of the existing X-tree structure highlighted above, motivates our pursuit for a solution that defeats the problems.

D. Packing X-tree (aX-tree construction)

Packing algorithms should be devoid of intersections between spatial objects or between the object and the walls of the container. In some variations, the focus is to find an ideal setting that packs a single vessel with the maximum density. On the other hand, according to Lodi et al. (2002), the aim could be to fill all the elements into as few containers as possible. Likewise, in a considerable number of modifications, the overlapping of the elements with each other or with the boundary of the vessel, is permitted, but should be minimized. We implemented an STR partitioning algorithm on X-tree on a 2D plane. The model is efficient for points and extended objects (lines and regions) query. Extended objects are stored by simply approximating their geometry using any one of two (2) different approaches described in Lee et al., (1996, September) and Dolci et al., (2010):

1. Using the object's MBRs (smallest axis aligned rectangle enclosing a spatial object), enclosed with the points (***x-min, x-max, y-min, y-max***), so as to estimate their extent in space using their minimum and maximum values as a single measurement on each axis
2. Using a more precise object decomposition methodology in cases of complex spatial objects, to brake the object down into simpler and smaller spatial components.

Note: for this project, we adopted the first approach.

E. Algorithm description and Pre-processing

The algorithm starts with the pre-processing, including refining the table and converting the *X/Y coordinate or LAT/LONG* columns to a standard geometric shape (polygons, points or lines) for SD, to get the extent. Following this stage, the algorithm continues by constructing an envelope (bounding box) around the object's extent. The next stage consists of calculating the midpoint of

the bounding box. The mid-points are sorted first by the *X-coordinate* and then ***β*** (total page- nodes - required for the data) is computed**. *R*** is the total objects in the database (rows), ***M*** is the maximum node capacity. In the following stage, **we begin the packing with the calculation ($\beta^{1/d}$), where *d* is the dimension.** This determines the total number of partitioning *J* required for the data space. Based on the result of the partitioning, the rectangles are loaded into pages (internal nodes) in groups of ***M*, with *rectangle ID, object ID*** and maximum node entry ***M*** as the input (that is, the algorithm take as input an array of pointers to a set of rectangles and a description of the maximum number of children for each node). This stage returns the **node/page ID**, and the MBR of the spatial objects.

We considered the centroids of the spatial objects rectangles (MBR) for indexing/ordering purpose, because the simple logic behind **a**X-tree indexing is:

- Filter the nodes by the first coordinate (*e.g. x-coordinate*)
- Filter the internal nodes on the subsequent coordinates (*e.g. y- and z- coordinates)*.
- After partitioning, scan the entire dataset and place each object in the right partition based on the underlying interval (range).

Figure 61 depicts the general structure of the **a**X-tree in 2-dimension. For improving the space performance of the algorithm, tree (3) important attribute *(rectangle id (ID), the object rectangle, and the Maximum node entry (**M**))*, makes up the storage of data objects in **a**X-tree. Those three (3) are the information necessary to differentiate between the data objects in the database. This mechanism guarantees a higher fan-out and a smaller internal node/directory (approximately 100% node fill), resulting in a better query performance and guarantees that the perimeter and area of the resulting MBRs is minimized. Since the fan-out of the tree is determined by the page size (that is, the size of the tree node matches the page size) of the external memory, and because we have assumed that each tree node consumes only one disk page on the disk storage (as such, where **M** is the size of a disk page). Therefore, each non-leaf node will contain ***M*** `number of children or at least` ≤ ***M***/2 number of children.

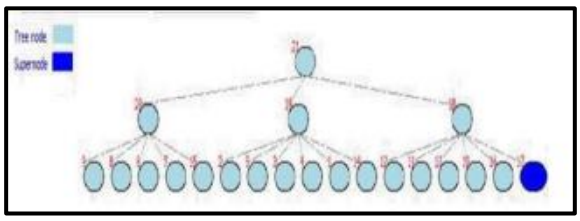


Figure 61: Typical aX-tree

We calculate the optimal value of the maximum node entry ***M*** to the tree as:

- Reduce size of database by storing only the significant information→ ***Φ***
- Typical size of the block storage (page) = 8kb for SQL server→Δ
- Let **β** = total nodes (page) required for the data
- **tR** = total rows or total database objects
- **T** = total nodes (Extent) required for the data

Then

$$\beta = \frac{\Phi(kb)}{\Delta(kb)}$$

$$\therefore M \cong \lceil \frac{tR}{T} \rceil \qquad \textbf{(L)}$$

Example:

Given a dataset with *0.500mb dataspace, 7313* data objects (rows), *8kb page size*, the size of

M will be: 7313/ (0.500*1000KB/7.800KB)

Otherwise, one could just in get the value of **T** as ***Φ (in mb) * 128*** (one extent in the block storage) for larger data and then get the value of ***M*** as **tR/ T**.

Instinctively, the calculation above assures a smaller directory/internal node, than other indexes at all time for ***d*** dimension.

`Note that for this experiment, the value of Δ is taken as 8kb – (96 bytes + 36 bytes + 78bytes) = 7.800kb.`

The idea is simple: we subtracted 96bytes for the page header, then 36bytes for row-offset and 78bytes free space on the page as typical of SQL server pages.

The idea of building the rectangles (MBR) around the objects (i.e. building the smallest enclosing block around them based on their value on the geometry or geography column of the table) is adopted to capture points and extended objects in a simple but efficient manner without having to create separate methods.

F. Partitioning and Bulk-loading algorithm

By applying the *STR* algorithm, the partitioning is rationally done, through a range (interval) partitioning strategy. Spatially adjacent/nearby objects are packed into one parent (internal) node. This guarantee minimized dead space in the parent MBR, densely filled with child MBRs. The entire data space is partitioned recursively, until all the dimensions (*x, y…N*) are considered.

The l*eaf node entry* is of the expression → (*oId, MBR*): *oId* (tuple) refers to individual elements in the database. The *MBR* is the slightest ***d*** dimensional bounding polygon around the data objects (for a *2d* - space, the MBR will be expressed in the form → x-min, y-min, x-max, y-max, and for *3d* space – x-min, y-min, z-min, x-max, y-max, z-max*). For non-leaf node entries, we have the tuple* → (*Cp, MBR, level*): where *Cp* is a pointer (child) to a lower level node and *MBR* is the rectangle encompassing *Cp* (which covers all the regions in the child node). Below, we have presented the three (3) main algorithms, created in this project.

Algorithm 6 (Procedure 1): partitioning

Input → (***Obj ID, geometric col***)
Build *bounding box* (***bb***) *of objects as*
    While ***R > 0***
    //Convert geometry column (geom) from table
        geom → ***Envelop*** *(xmin, xmax, ymin, ymax)*
Compute *Midpoint of bb*
    Midpoint → *([xmax – xmin], [ymax – ymin])/2*
Sort *rectangles (objects bounding box)*
    // Use value x-coordinate
Partition *sorted rectangles into* ***r*** → $\beta^{1/2}$ *groups of vertical slices (partition)*
    // β *is explained later //for* ***d*** *>2,* ***r*** → $\beta^{1/d}$
Sort r *on the y – coordinate of the rectangles center.*
Repeat 1 to 5 for each selected dimension
Load r *rectangles into nodes (pages),*
Output → *(MBR,* ***Node ID****), for each leaf level node that loaded into a temporary file to be processed in phase two of the* ***aX-tree*** *algorithm*

Procedure **2** starts with the temporary file resulting from Procedure **1**. At this stage, we build the **aX-tree** recursively moving upwards (starting from the leaves nodes), until the root node is constructed.

Algorithm 7 (Procedure 2): bulk-loading

```
Create leaf nodes → the base level (L = 0)
        While R /* in procedure 1*/ > 0
Create a new aX-tree node,
Allocate M rectangles (of R) to this node
        /* during node creation avoid overlapping nodes, extend to super-node in the current level (only for
        leave level) see algorithm 3 */
Create nodes at higher level (L + 1)
While (nodes at level L > 1)
Sort nodes at level L ≥0 on ascending creation time
Repeat
Return Root
```

An appropriate decision to extend a node to super-node, is logically made by the simple heuristics in procedure **3**.

Algorithm 5 (Procedure 3): Extend a Node (create super-node)

```
While creating the nodes in procedure 2
Do
        Consider total number of MBRs in each partition
        Consider the value of M
                For each partition
                        IF remaining RECTANGLE <= (M + ((M / 2) - 1))
                        &&      RECTANGLE > M
                        &&      RECTANGLE != 0
                //RECTANGLE is used to represents objects that falls into individual
        partitions Create S (maximum number of entry for super-node)
        {
        S =M*2; M = S;
        }
END DO
```

G. ANALYSIS:

The Performance of any search algorithm, when the underlying data is indexed with a hierarchical structure (e.g. tree), is evaluated by the amount of reads (***disk accesses)*** required for finding the desired object(s) in the database. Thus, the branching factor of the structure is selected so that the size of a node equals the size of a disk block (or a multiple of it) or the size of page (file system).

In some applications involving a database with high-dimensional data, NN queries are usually vital (Berchtold et al., 1996) thus, the major priority for NN exploration for databases sorted with a tree data structure is the ***CPU-time rate.***

A. Method Evaluation:

According to **Ester et al., (1999); Lazarevic et al., (2000),** the more algorithms become complex, and the more suggestions of hybrid methods, for combining a number of approaches emerge, it becomes exceedingly time consuming to implement these algorithms from scratch. Moreover, there is always a common DM algorithm suitable over all application domains. Increasing the robustness of DM systems, might involve the integration of DM architectures, thereby generating a system that applies different types of algorithms or hybrid techniques to handling a given dataset. Consequently, a prevalent architecture with several algorithms, collated into a package with the greatest suitability and facility for solving a targeted problem is chosen. Therefore, current research in SDs (**Schiller (2004)** aims mainly to improve their ***extensibility, functionality, and performance***. The motivation to improve functionality arises from the overwhelming requirements of several systems such as GISs, sensor networks and locality-based services. **Li and Wang (2005)** pointed out in support of the above proposition, the fact that the implementation of DM in spatial databases still requires further studies. **Shekhar et al (2011**) noted that recent research advances, in addition to the ever-increasing necessity for awareness of spatial information, has led to numerous commercial SDBMs including:

- **MLC++:** developed by **Kohavi et al. 1997),** gives users a good framework for designing different algorithms.
- **Geo-Miner:** a spatial adjustment of the classical DM application DBMiner
- **S-PLUS: an interface for a** software package (ArcView GIS) that offers tools for analysing distinct categories of S (e.g. geo-statistical data, spatial point patterns)
- **ESRI's Geodatabase (ArcGIS)**
- **Microsoft's SQL Server 2008**

Regardless of technologies listed above, **Tian et al, (2015)** ascertained that the present opportunities available to SDM systems, relational databases and **SDs** are not effectual in spatiotemporal **DMGT** because classical databases create a **B-tree index** on each table column. To curtail these limitations, the authors proposed a new spatiotemporal database constructed in addition to a traditional database, which is a new method for storing data by and implementing an R-tree, for faster data retrieval. The authors proposed an enhancement of the R-tree performance

by applying a space-filling curve (which builds a Morton R-tree) to the construction of the tree structure. However, in order to overcome the challenges faced by recent SDBM systems (scalability, inconsistency, data loss, and uncertainty), the method in **Tians et al (2015) described above,** needs further improvement.

Thus, we propose the implementation of **an improved X-Tree data** structure for the management and manipulation of SD, which is used on the layered architecture (detailed in section 2.2) to integrate both the relational/traditional database and the SD to build the complete system.

X-tree (The *e-Xtended node tree)* is a tree indexing structure built on R-tree for saving data that comes with several dimensions. The X-Tree according **Berchtold et al, (1996)** differs from R-trees and its variants because it lays great emphasis on preclusion of overlap in ***the internal nodes***, which develops into an issue in high dimensional. For situations where nodes cannot be further split without inhibiting overlap, node-split is delayed, leading to **super-nodes**. The tree linearizes in extreme cases, which covers for worst-case behaviours/scenarios observed in many data structures.

B. Time and Space complexity:

In terms of space and time taken to process a query (Arya et al. 1994, Cazal et al. 2013), for dimensions higher than 2, it is quite difficult to find algorithms that are worst case efficient. Nevertheless, for the **a**X-tree (our proposed structure), the capacity of every node is restricted to the value of the maximum number of entries (MBR or rectangles calculated in equation (**L**)), obtained by calculating the amount of available space spanning from 4kb page block size. Thus, for an optimal, efficient and cost effective I/O operation, every **a**X-tree internal node must occupy only a single disk page, giving us the number of pages at leave (or total leave nodes).

The **a**X-tree combats the worst-case scenarios of the existing X-tree listed below

(1) The structure degenerates to a linear array of the entire dataset when the full dataset is placed in one node (the root becomes a super-node),

(2) For the cases of zero super node, the tree becomes similar to an R-tree, where all the nodes are arranged hierarchically,

In case **(1),** even if the **a**X-tree has all its data in one node (which is very unlikely to occur, since the tree is loaded bottom up from the leave, based on the value of the capacity of each internal node calculated in equation **(L)**), it would not degenerate to a linear scan beyond the leave level. Because the worst that will happen is to have a large supernode plus other smaller internal nodes created at the leave.

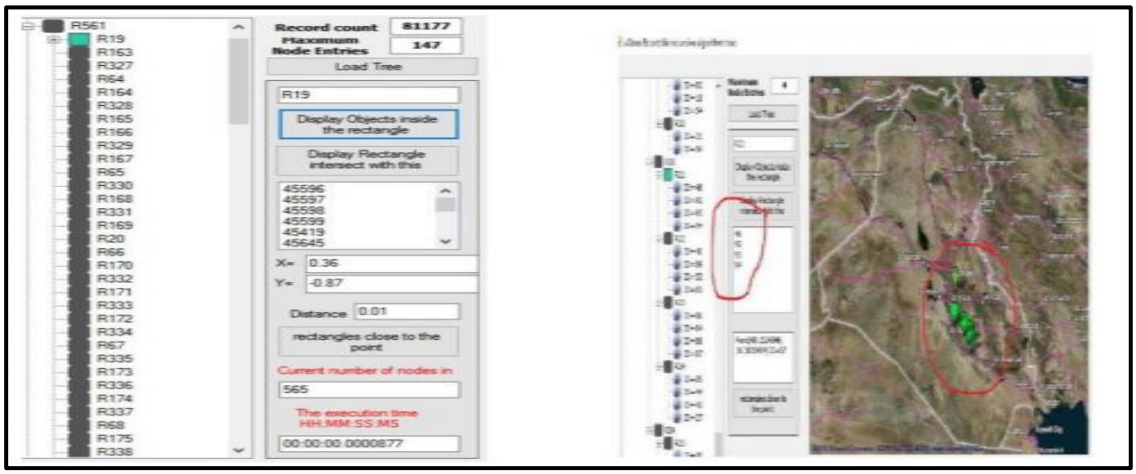
In case **(2),** for zero supernodes, **a**X-tree still offers a better performance against the X-tree with little or no overlap. This is achieved by making sure the internal nodes contain an equal predetermined number of objects and if there is a left over, then the last internal node contains lesser data.

Similarly, even with super-nodes (quite few) an overlap minimal partitioning is still maintained, because the super-node is only constructed if there is a spill over in the partitioning. The only thing similar between the two structures (**aX**-tree and X-tree) is the height of the tree, which behaves in the same way in the presence of an increased number of super-nodes by forcing a reduction in the tree height. The super-node is constructed so that after partitioning (by grouping objects according to the maximum node entries of the internal node), if the last collection is less than the least permitted, then the last node in the partition is extended to a super-node. Constructing the super-node in this fashion is meant to handle the non-uniformity and high skewness frequently present in distributed data (typical of spatial data). For homogeneously distributed data, the rectangles are assured to contain the same quantity of data, skewed datasets may vary by partition.

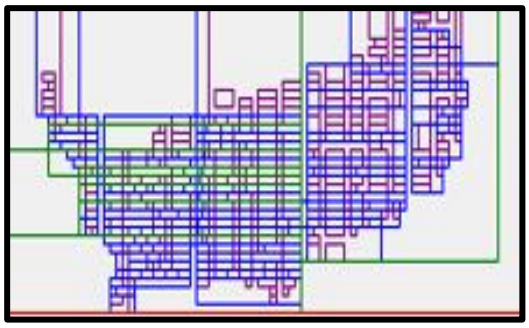
C. *Basic operation:*

The proposed algorithm is able to process a variety of queries, NN, containment, point, join (intersection) and range queries rapidly and efficiently. For example, Figure 62 (a) shows the time (00:00:00.0000877µs) to process a NN query on a database containing 81,177 polygon objects. One more interesting thing about **a**X-tree (proposed system) is the capability of predicting its performance a priori, based on the available disk page size. This means that space efficiency is achieved by predicting accurately, the total amount of space (calculated as total required pages in section 5.7 (E)) to be consumed and making necessary adjustments. In addition, another valuable aspect of the **a**X-tree, is that once the program is running (executed), if the table is on SQL server, increase in the table size on

any dimension, have no overbearing negative effect on the program.



(a)



(b)

Figure 62: Implementation of **a**X-tree algorithm on splitting a database with US postal districts, showing **(a**) the time taken to index and query 81,177 polygon objects **(b**) the output of the partitioning method.

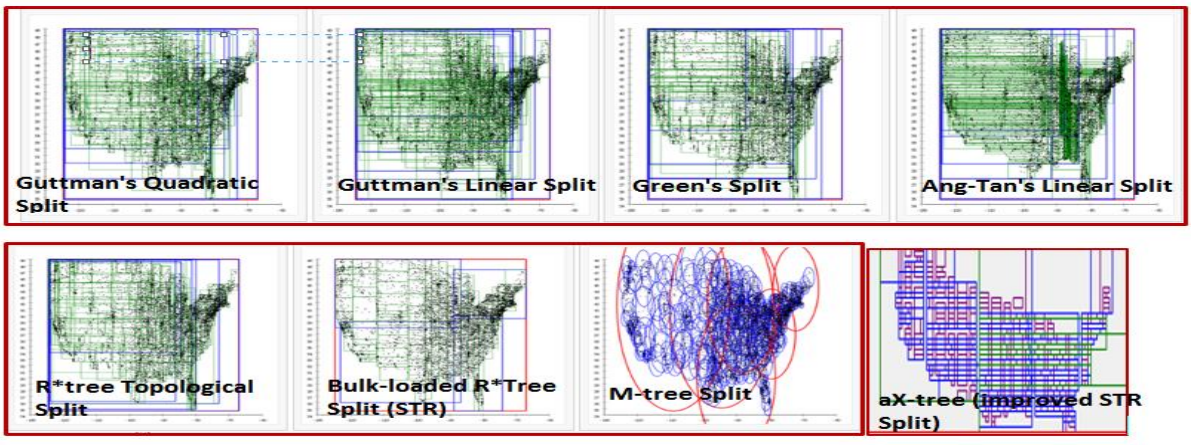


Figure 63: Comparison between aX-tree algorithm performance and other splitting algorithms on US postal districts database

D. *Performance:*

X-tree permits the super-node to grow several times bigger than a normal node size. The X+-tree on the other hand permits the super-node grow to at the maximum, the a normal node size multiplied by a known user value (known as MAX_X_SNODE) and eventually split into two new nodes when the super-node grows bigger than the upper limit, these scenarios can lead to deteriorated performance. Thus, in our proposed model, we limited the size of the super-node to just two (2) times the normal node size, such that the leave centric information (data contain on the leave nodes) is a direct consideration of the value M (see equation **(L))** for each partition. To improve **a**X-tree further, we restricted the super-node only at the leaf level (Figure 61). With this improvement, there is an increase in the speed efficiency of the structure since the block reading of the internal/directory ends at the index level.

Increasing the number/size of super-node helps the height (corresponding to the amount of page accesses compulsory for point queries) of the X-tree to reduce, with multiplying dimension. However, for the **a**X-tree, we are guaranteed that the height of the tree only grows exponentially only at $\log_M(R)$, which improves performance as the height of the tree is a function on M and M is optimized for space and speed efficiency.

Figure 62 depicts the potentials of our algorithm on a NN query for computing the closest rectangle (MBR) to a query point from a database of 81,177 polygon objects, which represents the different towns within our study area. Figure 62(b) is the screenshot of the packed rectangles in purple, the index node in blue and the directory nodes in green, while the root is in red rectangle respectively. Figure shows a comparison of our proposed structure to other spatial indexing structures.

E. *Conclusion:*

In this project, we looked at spatial indexing for managing large SDs. The study shows that the X-tree (optimised for dynamic environment) performs exceptionally well, in terms of spatial data indexing for efficient query performance. Nevertheless, the X- tree begins to deteriorate as regards space for very large databases. Therefore, we focused on investigating the possibilities of improving the X-tree. As such, we propose an adjusted/packed X-tree (**a**X-tree), a method for building a packed X-tree by bulk loading the existing X-tree structure. Due to the mode of construction, the **a**X -tree performs better

than the existing structure as regards speed and space management. Rather than inserting each object iteratively (one after the other) into an initially empty tree, our proposed structure permits pre-processing of data before packing. Packing methods constructs a tree at a time, since the underlying data is already at hand. By ensuring maximal/efficient node occupancy, a good bulk loading method will always construct fast for stationary objects with a guarantee of reduced total of wasted empty spaces on the tree pages.

The proposed **a**X-tree performs faster tree loading for the whole spatial objects at once, thus, decreasing redundant spaces in the index nodes of the tree. This method leads to producing a better splitting of spatial objects into the tree nodes.

Because the super-node in **a**X-tree is leave centric, the structure has the benefit of

(1)) Minimal tree height

(2) High quality of directory node

(3) Least overlap of bounding boxes of internal nodes

(4) Reduced area of rectangle (MBR)

(5) Maximized space efficiency.

CHAPTER 6. Result and Discussion:

6.1 Discussion:

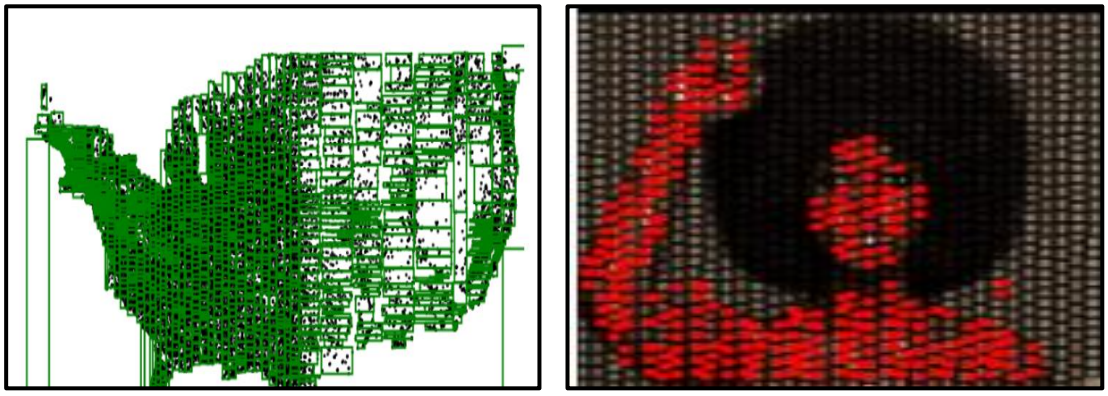
In this project work, we investigated ways of improving the efficiency of data retrieval and information extraction from large datasets. The work is a prodigy of indexing systems for spatial-biased datasets. Given any dataset, we start by extracting their spatial quantities (explained in details in section 3.2.1), to apply S tools to mining information from the data. Using the information extracted from the data, we make predictions or recommendations based on the outcome of our analyses. By so doing, we can perform any DM/ML task (prediction/classification, grouping/clustering exploration etc.) on the dataset. The investigation involves learning about existing methods for efficient patterns using spatial data tools and techniques. As we saw in chapter 2, spatial data generally describes any type of data where the location of the data object in space (see more about space in chapter 3) holds importance for mining purposes. Many methods exist for mining information from spatial. However, from the outcome of our analyses of the existing method (see section 1.4); we are of the opinion that new efficient methods are still needed.

Therefore, as our contribution, we presented the packed maintained k-dimensional tree *(Pmkd-tree* discussed in section 5.5) and we also implemented a novel packed quad-tree (Pquad-Tree), both structure of which to speed up the performances of tree structure for efficient indexing, querying, retrieval, learning, mining and knowledge representation. As seen in the result in section 5.5.8, the proposed spatial structure performs better with very low false positive/negative rate as compared to many latest systems.

The key underlying framework behind this system is the non-overlap efficient partitioning algorithm applied in building the *Pmkd-tree* structure. The partitioning algorithm works similar to STR method proposed in Leutenegger et al (1997), however with difference that the sort-tilt-tile (sTT) is in-memory based and applies efficiently to static datasets. The fundamental power of the sTT partitioning algorithm lies in the fact that, its performance is not affected by the data i.e. given a dataset of size $\boldsymbol{n}$, and another dataset of size $\boldsymbol{m}$, where $\boldsymbol{m} > \boldsymbol{n}$, it takes equal time for both sizes of data to be partitioned. Rather, the total partitions and size of each partition is determined by $\boldsymbol{\mu=25}$. Secondly, the partitioning strategy can apply efficiently to any learning task, classification, clustering, NN etc.

The partition strategy for two (dimension) is overlap free, shallow depth and space efficient. The sTT algorithm can apply to spatial/non-spatial datasets and works efficiently on both cases. Looking at the result of the partitioning algorithm in Figure 64 (a) and Figure 64 (b), one can see

that the sTT partitioning algorithm highly overlap free. This means that time complexity for search procedure on data indexed with a structure that applies sTT partition = **O (log n)**



(a) (b)

Figure 64: Result of applying the sTT algorithm for partitioning different kinds of data

6.2 Resources

Sources of data: For this project, the main information source include books, journals, articles, the internet and other similar existing materials on building algorithms. We examined existing and similar advanced projects done using different methodologies and techniques. For the experiments, free available relevant online datasets were used. For instance, we use images from We used images from the ColorFERET dataset NIST (2019), Pratheepan human skin dataset Tan et al, (IEEE T-II, 2012) and various other image of diverse complexion, pose, orientation, age, variation of illumination and sex, selected from the internet. Dataset from other databases consisting of non-spatial data, which formed the database of the iris plant dataset from UCI machine-learning-databases website (UCI, 1998) was also used. As explained in section 4.2, we used the multidimensional scaling (MDS) for pre-processing, to add spatial referencing to the dataset by projecting the multidimensional data on a two-dimensional Cartesian coordinate planeR^2 (we explained the Cartesian coordinate system in chapter 3).

Hardware: First, the computer system is the hardware prerequisite for our project, every other data, processes, implementation and information would depend on it.

Software: For this project, various software tools were employed. For the database and data storage, **we used SQL server, excel, and some other direct web content.** Statistical tool **SPSS** was employed for carrying out statistical analysis. For writing the codes, C++, Python and

JavaScript was used. Most of our codes were self-written and no open source tool was used for coding purposes.

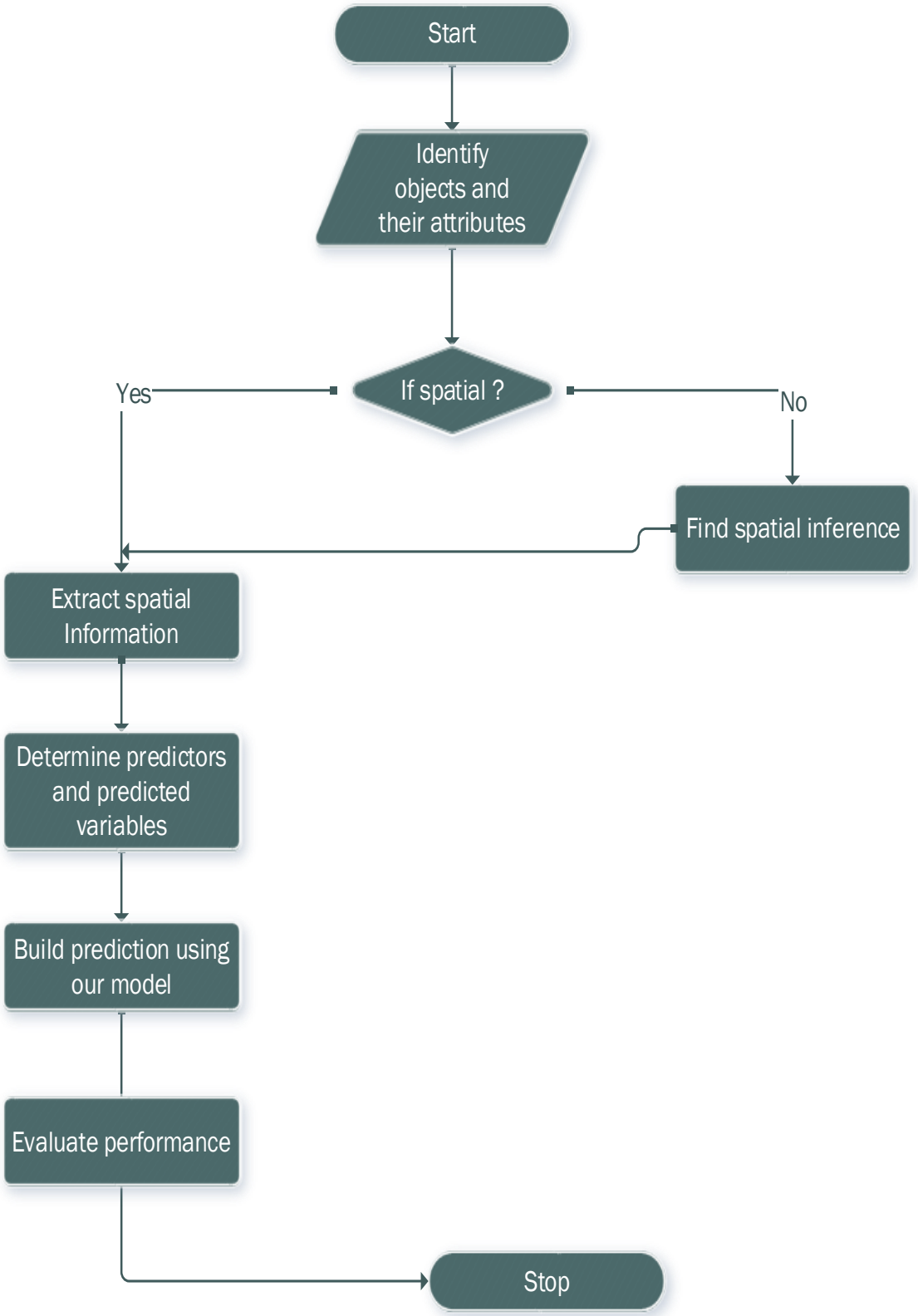
6.3 General Analysis of System Requirements

In this research work/study, we considered various non-functional requirements that are essential for spatial data analysis. These requirements are listed as below:

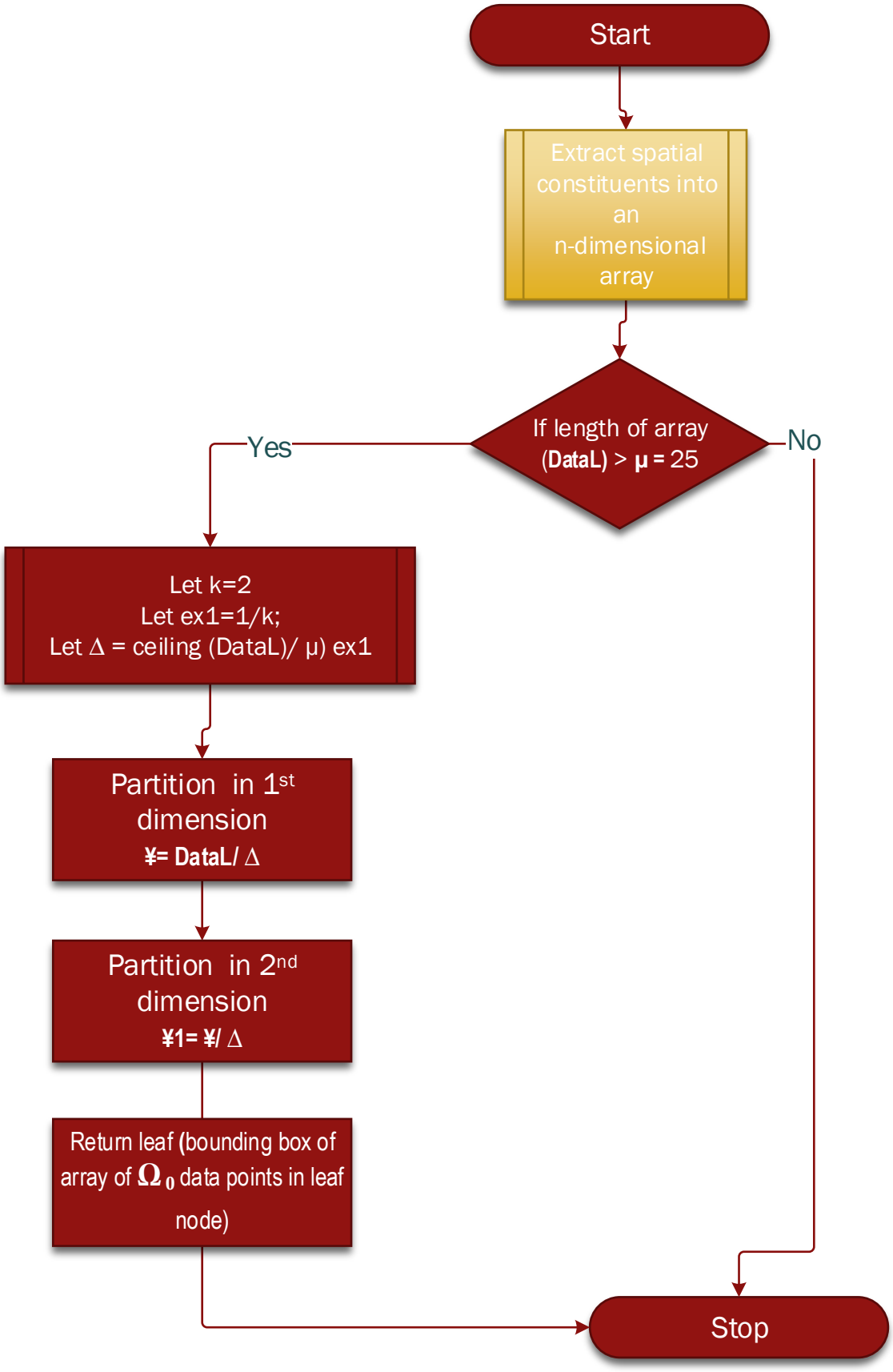
- The physical environment (***non-spatial objects, multiple sites, event scenes, as individual and prospective or potential environments*** etc.).
- Interfaces (***interaction medium, user-friendly system consideration*** etc.).
- Some human or physical factors (***that is variables would represent spatial characteristics of spatial factors, and how do we extract the qualities and quantities from varying dataset***).
- The measure of system performance ***(this consideration measures how well the algorithm or model functions as regards predictions and other ML and DM tasks***).
- Data (***spatial qualities and qualitative substance***).
- Resources (***deriving spatial information from a given dataset, finding, physical space***).

Flowchart 1- Phase 1: Spatial modelling (objects in space) procedures

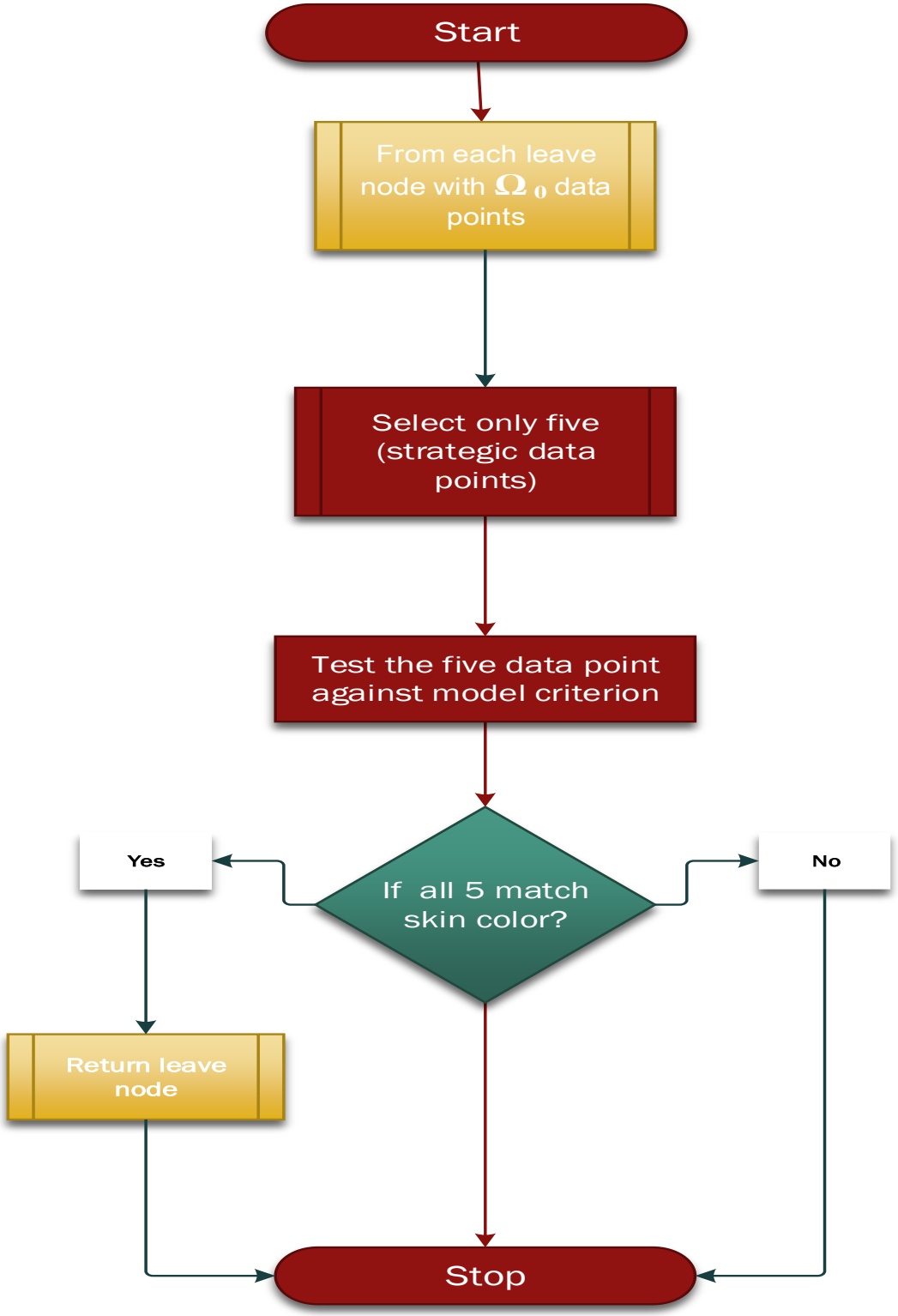
Flowchart 1 depicts the various stages involved in the spatial modelling process including data acquisition, exploration and analysis, spatial quantity extraction and model application.


Start
Identify objects and their attributes
If spatial ?
Yes
No
Find spatial inference
Extract spatial Information
Determine predictors and predicted variables
Build prediction using our model
Evaluate performance
Stop

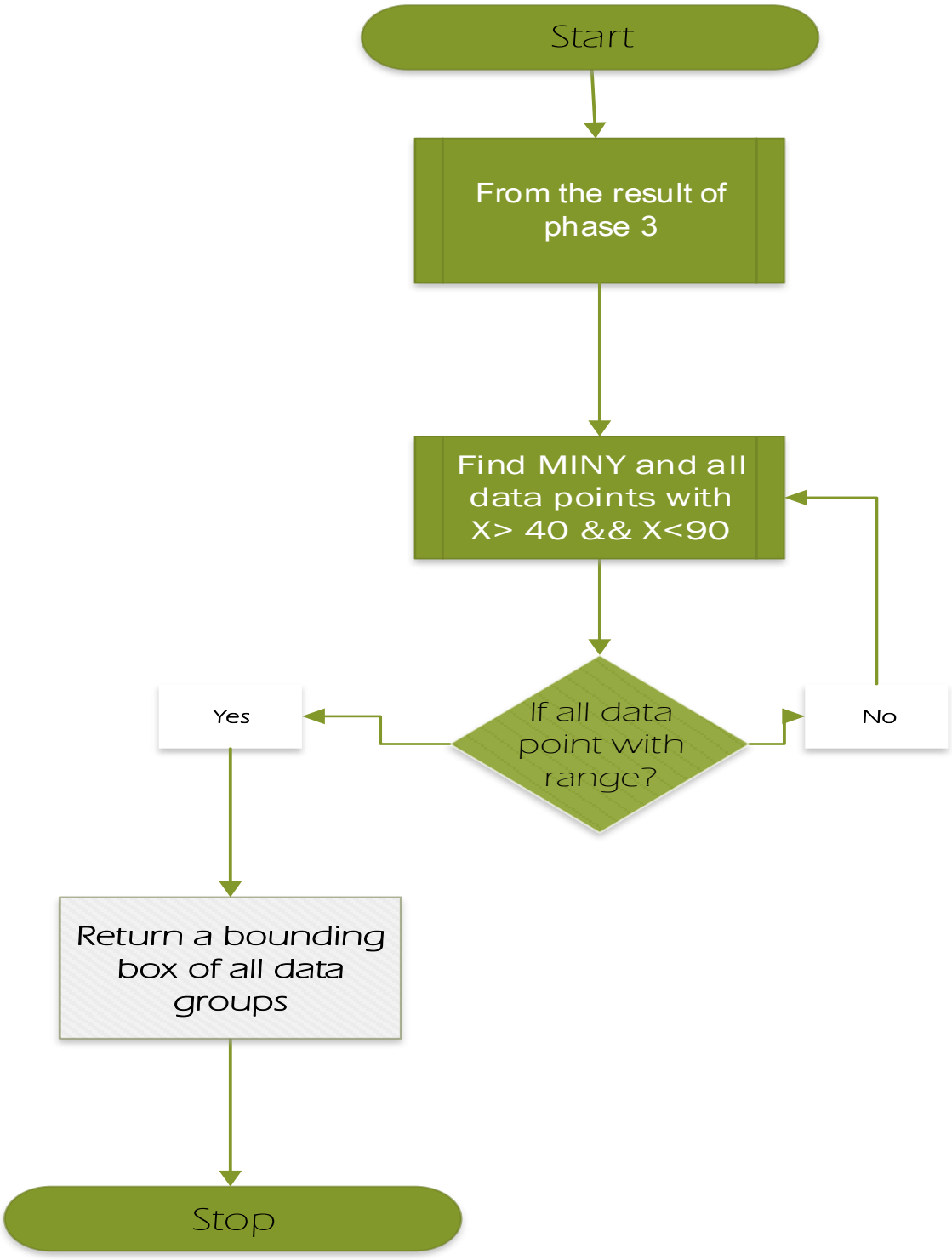
Flowchart 2- Phase 2: Partitioning procedures



Flowchart 3 - Phase 3: Skin detection procedures



Flowchart 4 - Phase 4: Finding face with our clustering technique



CHAPTER 7. CONCLUSION:

Objects like medical or satellite image data, geographic data, maps and very large-scale integration (VLSI), biometric and human related data, computer-aided design data or any other form of object, in one way or the other contain spatial-related information. To extract their spatial integrant, these objects may be represented on the computer in either

1) **Raster format**, consisting of *k-dimensional* bit or pixel maps. For instance, a 2D satellite image can possibly be characterized as raster data, with each pixel of the objects' image, representing the rainfall or temperature in a given area.

2) **Vector format**, whereby objects like roads, buildings, bridges and ponds are characterised as overlays or unions of elementary geometric paradigms, including networks (partitions), lines, and points moulded by these components.

Due to the huge progress in the production, dissemination, distribution and application of large computerized spatial data (obtained from the objects mentioned above), the acquisition and storage processes of these data results in the growth of huge spatial databases. As such, we are confronted with huge amounts of increasing spatial data; which means that an end user might face more difficulty in understanding them without the helpful knowledge interpretation from efficient spatial databases.

Traditional/classical database systems are designed for business, administrative, and financial applications. In most cases, they deal exclusively with certain standards, that is, numeric data including integers, dates and floats in addition to alphabetic data like strings. Object-oriented and object-relational DSs are developed as powerful mechanisms for the management of big volumes of this kind of data in an effective manner. Nevertheless, more of similar data, otherwise known as non-standard data, arise from new, evolving applications and new application areas. Typical examples include image data, video data, genomic data, multimedia data, temporal data, medical data, spatiotemporal and spatial data, to mention only a few. Two (2) critical features of these non-standard data contrary to alphanumeric data are;

(1) Possession of an inner, complex structure

(2) Possibly arbitrary, with finite representation length.

Furthermore, domain-specific knowledge is crucial in order to support these non-standard DSs. Classical database schemes have huge complications coping with these kinds of data owing to the fact that they have been set up to fixed-length data of very basic internal structure.

In this project, I have created some advanced, high performance algorithms that can be used to query a spatial database efficiently overcoming all the above-mentioned challenges using the new (tree-based) data structures**.**

All our proposed models (packed maintained k-dimensional tree --***Pmkd-tree***, packed quad-tree --***Pquad-Tree***, adjusted X-tree --***aX-tree***, parallel adjusted X-tree DBSCAN --***paX-DBSCAN*, and** *adjusted X-tree on DBSCAN – **XDBSCAN***) have been described extensively in this research work. Each of them is described fully and a link to the full content is provided. I have implemented and evaluated all the models using several forms of datasets (simulated and real world) and the results demonstrate that these models perform better than many state-of-the-art.

The problem or drawback of most tree-based structures is that they are prone to so much superfluous **search of data MBR, owing to the high intersection between the MBRs of the internal tree nodes**. Using our new split technique/algorithm, the overlap minimal split typical features of the structure produces an improved performance.

For each project in this research work, we evaluated the result by testing the model against existing models. In each case, real world and synthetic data was used. The models were tested to ascertain their efficiency. The results from each project shows that they perform better than existing methods. For instance, the skin and ace detection algorithms are characterised by high precision and accuracy rates, with very low false hit. The partitioning strategy is quite optimal and guarantees an overlap free partitioning in the directory nodes. Our clustering algorithm on the other hand is quite efficient, and applies to fast retrieval procedure for finding human faces in the image as discussed in section 5.6. The final model is general purpose and could apply to various DM and ML tasks.

Note: All the published work was peer reviewed.

Finally, following the general hypothesis of this research project in section 1.2, we hereby claim that

1. With our main model, given any form of dataset, a new spatial inferential rule is discovered efficiently and rapidly.
2. Given any form of data, indexed with our proposed model, for dimension $(n) = 2$ in R^n space complexity will reduce significantly, since data is pre-processed and the packing method decides space/storage usage a priori
3. *At all point in time, even if the* degree of overlap *(dOp)* between instances *of the data increases, the speed of query will not be affected adversely, if our model is applied*

5.2. **FUTURE WORK:** Our future work will mainly focus on optimizing the proposed systems especially for large non-spatial data (big data), because presently, we have not applied any of the methods to non-spatial datasets. Additionally, we shall implement the algorithm using a new framework to reduce the rigidity of the system in terms of the pre-processing of data input. The distance measure employed by our model at the moment is the Euclidean distance, so we will look at the performance of the models with other distance metrics (Manhattan, theoretical, Chebyshev) and compare the various results. In addition, we shall optimize the systems for stream data, as most data in recent times are continuous. Presently, NN queries only return the value of the closest neighbour or all the neighbours within a certain distance to the query object, so these structures shall be expanded to capture the ***kth***-NN. To make ***Pmkd-tree faster***, we aim to parallelize the tree construction. Again, for the face recognition program, the structure still finds it a bit difficult when there are more than one human face on the given picture or image, so we shall look into ways of improving the structure performance in that dimension.

Finally, the main goal/objective of this study/program is to produce a widely accepted application for solving some of the current real-world problems that depend on advanced computational methods using spatial DM techniques. Therefore, the most important upgrade to the structures proposed here is the integration of the ***Pmkd-tree*** and the ***Pquad-Tree***, this venture we believe will make a great difference as compared to implementing the structure individually. Consequently, we look to produce an algorithm for **pattern recognition** that is very effective, accurate, precise, fast, and most importantly highly regarded in the computing field.

REFERENCES:

1. Abbas, A. R., & Farooq, A. O. (2018, October). Human Skin Colour Detection Using Bayesian Rough Decision Tree. *In Proc. International Conference on New Trends in Information and Communications Technology Applications*. (pp. 240-254). Springer, Cham.

2. Adam, A., Patricia, B., Brian, D., Alison, E., Erik, G., Luebering, J.E., Amy, M., Melissa, P., John, P. R., Michael, R., Kara, R., Amy, T., Jeff, W., Adam, Z., and Alicja Z. (2008). "Space – Physics and Metaphysics". Encyclopaedia Britannica. https://www.britannica.com/science/space-physics-and-metaphysics. Accessed 4th May 2018.

3. Agrawal, R, T Imielinski and A Swami (1993). Mining association rules between sets of items in large databases. In ACM SIGMOD International Conference on Management of Data, pp. 207–216.

4. Albiol, A., Torres, L., Delp, E. (2001). Optimum colour spaces for skin detection. *In Proc. International Conference on Image Processing (ICIP).* (pp. 122- 124).

5. Ali, A. A., El-Hafeez, T. A., & Mohany, Y. K. (2018). A Robust and Efficient System to Detect Human Faces Based on Facial Features. *Asian Journal of Research in Computer Science*, 1(12).

6. Ajiand, A. & Wang, F. (2016). Big Data: Storage, Sharing, and Security. Eds: Fei Hu. Taylor & Francis LLC, CRC Press.
7. Ajit, S. and Deepak, G. (2011) "Implementation and Performance Analysis of Exponential Tree Sorting" *International Journal of Computer Applications* ISBN: 978-93-80752-86-3 24 (3) pp. 34-38.

8. Alkathiri, M., Abdul, J. and Potdar, M. B. (2016). Geo-spatial Big Data Mining Techniques. International Journal of Computer Applications 135(11):28-36.

9. Amirbekyan, A., and Estivill-Castro, V. (2006). Privacy preserving DBSCAN for vertically partitioned data.
10. In Intelligence and Security Informatics (pp. 141-153). Springer Berlin Heidelberg.

11. Arya, S., Mount, D. M., Netanyahu, N., Silverman, R., & Wu, A. Y. (1994, January). An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In Proc. 5th ACM-SIAM Sympos. Discrete Algorithms (pp. 573-582).

12. Assent, I. (2012). Clustering high dimensional data. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(4), 340-350.

13. Banerjee, S., Carlin, B. P., & Gelfand, A. E. (2014). *Hierarchical modeling and analysis for spatial data*. Crc Press.

14. Ban, Y., Kim, S. K., Kim, S., Toh, K. A., and Lee, S. (2014). Face detection based on skin colour likelihood. *Pattern Recognition*, 47(4), pp. 1573-1585

15. Bayer, R. and McCreight E. (1972) "Organization and Maintenance of Large Ordered Indexes", Acta Informatica, 1(3), pp. 173–189,

16. Baskan, S., Bulut, M. M., & Atalay, V. (2002). Projection based method for segmentation of human face and its evaluation. *Pattern Recognition Letters, 23*(14), pp. 1623-1629.

*17. Bentley, J. L. (1975). "Multidimensional binary search trees used for associative searching". Communications of the ACM. **18** (9): 509. doi:10.1145/361002.361007.*

18. Berchtold, S., Keim, D. A.., Kriegel, and HansPeter (1996). "The X-tree: An Index Structure for High-Dimensional Data". Proceedings of the 22nd VLDB Conference (Mumbai, India): 28– 39.

19. Berchtold, S., Keim, D. A., & Kriegel, H. P. (2001). An index structure for high-dimensional data. *Readings in multimedia computing and networking*, *451*.

20. Berkhin, P. (2006). A survey of clustering data mining techniques. In Grouping multidimensional data (pp.25-71). Springer Berlin Heidelberg.

21. Bhattacharya A. (2014) Fundamentals of Database Indexing and Searching. Chapman and Hall/CRC. *ISBN 9781466582545*

22. Bhosale, H. S., & Gadekar, D. P. (2014). A REVIEW PAPER ON BIG DATA AND HADOOP. International Journal of Scientific and Research Publications, 756

23. Bickel, P., Diggle, P., Fienberg, S., Gather, U., Olkin, I., & Zeger, S. (2001). Springer Series in Statistics.

24. Bijuraj, L. V. (2013). Clustering and its Applications. In Proceedings of National Conference on New Horizons in IT-NCNHIT (p. 169).

25. Billings, S. (2013). *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains* (1st ed.). Hoboken: Wiley.

26. Bin Li, Lihong Shi, Jiping Liu and Liang Wang (2012) research on Spatial Data Mining in E-Government Information System " Data Mining Aplication in Engineering and Medicine" edited by Adem Karahoca, ISBN 978-953-51-0720-0, 336pages, Publisher: in-Tech, Chapters published August 29, 2012 under CC By 3.0 licence

27. Bouveyron, C., Girard, S., & Schmid, C. (2007). High-dimensional data clustering. Computational Statistics& Data Analysis, 52(1), 502-519.

28. Bogorny, V., Kuijpers, B., & Alvares, L. O. (2008). A Spatio-temporal Data Mining Query Language for Moving Object Trajectories. *Instituto de Informatica, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil Technical Report TR-357, 22*.

29. Boinski, P., & Zakrzewicz, M. (2014). Algorithms for spatial collocation pattern mining in a limited memory environment: A summary of results. *Journal of Intelligent Information Systems, 43*(1), 147-182. doi:10.1007/s10844-014-0311-x

30. Brakatsoulas, S., Pfoser, D., & Tryfona, N. (2004). Modeling, storing and mining moving object databases. Paper presented at the 68-77. doi:10.1109/IDEAS.2004.1319779

31. Buolamwini, J. and Gebru, T. (2018). "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification". Proceedings of Machine Learning Research, vol. 81. pp. 1–15.

32. Bush, I. J., Abiyev, R., Ma'aitah, M. K. S., & Altıparmak, H. (2018). Integrated artificial intelligence algorithm for skin detection. *In Proc. ITM Web of conferences* ((16), p. 02004). EDP Sciences.

33. Candan, K. S. and Sapino, M. L. (2010). Data management for multimedia retrieval. Cambridge University Press.

34. Cam D.P. (2012) Feature Space. Stack exchange. Online: Available at < https://stats.stackexchange.com/questions/46425/what-is-feature-space>. Accessed 4th May 2018

35. Carlson, K.(2012, August) "Difference between 'space' and 'mathematical structure'". Stack exchange:
36. https://math.stackexchange.com/questions/177937/difference-between-space-and-mathematical-structure. Accessed 3rd May, 2019.

37. Cazals, F., Emiris, I. Z., Chazal, F., Gärtner, B., Lammersen, C., Giesen, J., and Rote, G. (2013). "D2. 1: Handling HighDimensional Data". Computational Geometric Learning (CGL) Technical Report No.: CGL-TR-01.

38. Chakrabarti, K., & Mehrotra, S. (1999) "The hybrid tree: An index structure for high dimensional feature
39. spaces". In Data Engineering, 1999. Proceedings. 15th International Conference on (pp. 440-447). IEEE.

40. Chakrawarty, L., & Gupta, P. (2014) "Applying SR-Tree technique in DBSCAN clustering algorithm"
41. International Journal of Application or Innovation in Engineering & Management (IJAIEM). ISSN 2319-4847. 3 (1) pp 207 -210

42. Chen, C., Lin, J., Wu, X., & Wu, J. (2015). Parallel and distributed spatial outlier mining in grid: Algorithm, design and application. *Journal of Grid Computing, 13*(2), 139-157. doi:10.1007/s10723-015-9326-y

43. Chen, L., & Brown, S. D. (2014). Use of a tree-structured hierarchical model for estimation of location and uncertainty in multivariate spatial data. *Journal of Chemometrics, 28*(6), 523-538. doi:10.1002/cem.2611

44. Chen, H. H., Ding, J. J., & Sheu, H. T. (2014). Image retrieval based on quadtree classified vector quantization. *Multimedia tools and applications*, *72*(2), 1961-1984.

45. Chen, J. Y., Chen, J., Yu and Yang Z. (2011). Comparisons with spatial autocorrelation and spatial association rule mining. *Proc. IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services. (*ICSDM'11), June 29–July 1, 2011, IEEE, pp. 32–37

46. Chen, W., Wang, K., Jiang, H., & Li, M. (2016). Skin colour modeling for face detection and segmentation: a review and a new approach. *Multimedia Tools and Applications, 75*(2), 839-862.

47. Chen, M., Gao, X., & Li, H. (2010, April). Parallel DBSCAN with priority r-tree. In *2010 2nd IEEE International Conference on Information Management and Engineering* (pp. 508-511). IEEE.

48. Da'San, M., Alqudah, A., & Debeir, O. (2015, May). Face detection using Viola and Jones method and neural networks. In *2015 International Conference on Information and Communication Technology Research (ICTRC)* (pp. 40-43). IEEE. doi: 10.1109/ICTRC.2015.7156416

49. Daniel A M Villela, Codeço, C. T., Figueiredo, F., Garcia, G. A., Maciel-de-Freitas, R., & Struchiner, C. J. (2015). A bayesian hierarchical model for estimation of abundance and spatial density of aedes aegypti: E0123794. *PLoS One, 10*(4) doi:10.1371/journal.pone.0123794 de Chile, Chile. September 12–15, 1994. Kaufmann, pp.487–499.

50. Densham P J, Goodchild M F, Spatial decision support systems: A research agenda, In: Proceedings GIS/LIS'89, Orlando, FL., 1989, pp. 707-716.

51. Deren, Li, Shuliang Wang, (2005) "Concepts, Principles And Applications Of Spatial Data Mining And Knowledge Discovery", ISSTM, August, 27-29, 2005, Beijing, China

52. Dash, J. Patra, D. & Pradhan C. (2015) "A Proposed Hybrid Spatial Indexing: QX Tree" *International Journal of Computer Science and Information Technologies*. 6 (2) pp. 1737-1739. ISBN:0975-9646

53. Dastane, T., Rao, V., Shenoy, K., & Vyavaharkar, D. (2018). An Effective Pixel-Wise Approach for Skin Colour Segmentation-Using Pixel Neighbourhood Technique. *International Journal on Recent and Innovation Trends in Computing and Communication*, *6*(3), pp. 182-186.

54. Densham, Paul & F. Goodchild, M. (1989). Spatial decision support system: a research agenda. GIS/LIS '89. Proc. annual conference, Orlando, 1989. Vol. 2.

55. Delbiaggio, N. (2017). A comparison of facial recognition's algorithms. Online at: <https://core.ac.uk/download/pdf/84801048.pdf: Accessed 19th March 2019>

56. Developers (2019). Classification: Accuracy. *Machine Learning Crash Course*. https://developers.google.com/machine-learning/crash-course/classification/accuracy. Accessed 4th April 2019.

57. Ding, Z., Yang, B., Chi, Y., & Guo, L. (2015). Enabling smart transportation systems: A parallel spatio-temporal database approach. *IEEE Transactions on Computers*, 1-1. doi:10.1109/TC.2015.2479596

58. Doja, M. N., Jain, S., and Alam, M. A. (2012) ―SAS: Implementation of scaled association rules on spatial multidimensional quantitative dataset. International Journal of Advanced Computer Science and Applications Vol. 3, (9) pp. 30-35

59. Dolci, C., Salvini, D., Schrattner, M. and Weibel R. (2010) "Spatial Partitioning and Indexing" *Geographic Information Technology Training Alliance (GITTA)*. Available at < http://www.gitta.info/SpatPartitio/en/text/SpatPartitio.pdf>. Accessed: 30 Dec. 2016.

60. Dwivedi, D. (2018). Face Detection for Beginners. <Online: Available at: https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9. Accessed 22nd March 2019>

61. Economides, G., Piskas, G. and Siozos-Drosos, S. (2013). Spatial Data and Hadoop Utilization.

62. Eldawy, A., & Mokbel, M. F. (2015, April). The era of big spatial data. In *2015 31st IEEE International Conference on Data Engineering Workshops* (pp. 42-49). IEEE.

63. Eldawy, A., Mokbel, M. F., Alharthi, S., Alzaidy, A., Tarek, K., & Ghani, S. (2015). SHAHED: A MapReduce-based system for querying and visualizing spatio-temporal satellite data. Paper presented at the 1585-1596. doi:10.1109/ICDE.2015.7113427

64. Eldawy, A., & Mokbel, M. F. (2015, June). The Era of Big Spatial Data: Challenges and Opportunities. In *2015 16th IEEE International Conference on Mobile Data Management* (Vol. 2, pp. 7-10). IEEE.

65. ElFkihi, S., Daoudi, M., & Aboutajdine, D. (2006, March). A tree distribution for skin detection. *In Proc. The Second International Symposium on Communications, Control and Signal Processing (ISCCSP'06)*.

66. Engineering (ICIME), 2010 The 2nd IEEE International Conference on. IEEE, 2010, pp. 508-511.

67. Enriquez, K. (2018). Faster face detection using Convolutional Neural Networks & the Viola-Jones algorithm. Online: Available at: <https://www.csustan.edu/sites/default/files/groups/University%20Honors%20Program/Journals/01_enriquez.pdf: Accessed 22nd March 2019.

68. Ernest, R., & Djaoen, S. (2015). *Introduction to SQL Server Spatial Data.* Retrieved from https://www.simple-talk.com/sql/t-sql-programming/introduction-to-sql-server-spatial-data.

69. Ester, M., Kriegel, H. P., & Sander, J. (1997). Spatial data mining: A database approach. In *Advances in spatial databases* (pp. 47–66). Springer Berlin Heidelberg. doi:10.1007/3-540-63238-7_24

70. Ester, M., Kriegel, H. P., & Sander, J. (1999). Knowledge discovery in spatial databases. In Mustererkennung 1999 (pp. 1-14). Springer Berlin Heidelberg.

71. Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, No. 34, pp. 226-231).

72. Fan, B. (2014). Hybrid spatial data mining methods for site selection of emergency response centers. *Natural Hazards, 70*(1), 643-656. doi:10.1007/s11069-013-0833-5

73. Fan, W., & Luo, W. (2009). The key technologies research of spatial data mining based on the GIS grid services. Paper presented at the 1-4. doi:10.1109/CISE.2009.5366396

74. Fowler, A. M., & Bridge, M. C. (2015). Mining the british isles oak tree-ring data set. part A: Rationale, data, software, and proof of concept. *Dendrochronologia, 35*, 24-33. doi:10.1016/j.dendro.2015.05.008

75. Giao, B. C., and Anh, D. T. (2015) "Improving Sort-Tile-Recusive algorithm for R-tree packing in indexing time series" In Computing & Communication Technologies-Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on (pp. 117-122). IEEE.

76. Grm, K., Štruc, V., Artiges, A., Caron, M., & Ekenel, H. K. (2017). Strengths and weaknesses of deep learning models for face recognition against image degradations. *IET Biometrics, 7*(1), 81-89.

77. Guo, D., & Cui, W. (2008). Mining moving objects trajectories in location-based services for spatio-temporal database update. *Proceedings of SPIE - the International Society for Optical Engineering, 7143* doi:10.1117/12.812625

78. Guttman A. (1984)" R-trees: A Dynamic Index Structure for Spatial Searching" Proc. ACM SIGMOD, pp. 47–57

79. Güting, R. H., & Schneider, M. (1993). Realms: A foundation for spatial data types in database systems. In Advances in Spatial Databases (pp. 14-35). Springer Berlin Heidelberg. doi:10.1007/3-540-56869-7_2 Hinneburg,

80. Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal—The International Journal on Very Large Data Bases*, *3*(4), 357-399.

81. Faria, R. A. D., & Hirata Jr, R. (2018). Combined Correlation Rules to Detect Skin based on Dynamic Colour Clustering. In *VISIGRAPP (5: VISAPP)* (pp. 309-316).

82. Hajari, H., & Hakimpour, F. (2014). A Spatial Data Model for Moving Object Databases. *arXiv preprint arXiv:1403.3304*.

83. Han, J., Li, Z., & Tang, L. A. (2010). Mining moving object, trajectory and traffic data. In *Database systems for advanced applications* (pp. 485-486). Springer Berlin Heidelberg.

84. Hahsler. M., Piekenbrock P., Arya, S. and Mount, D. (2016) Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms.<online available at> [https://cran.rproject.org/web/packages/dbscan/dbscan.pdf]

85. Haimowitz, I. J., & Kohane, I. S. (1993, August). Automated trend detection with alternate temporal hypotheses. In *IJCAI*(Vol. 93, pp. 146-51).

86. Hashavit, A., Levin, R., Guy, I., & Kutiel, G. (2016, July). Effective Trend Detection within a Dynamic Search Context. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (pp. 817-820). ACM.

87. Hassan, E., Hilal, A. R. and Basir, O. (2017). Using ga to optimize the explicitly defined skin regions for human skin colour detection. *In Proc. of the 30th IEEE Canadian Conference on Electrical and Computer Engineering, (CCECE 2017).* (pp. 1–4, 2017).

88. Huang, Y., & He, Z. (2015;2014;). Processing continuous K-nearest skyline query with uncertainty in spatio-temporal databases. *Journal of Intelligent Information Systems, 45*(2), 165-186. doi:10.1007/s10844-014-0344-1

89. Hung-Yi Lin, (2008) "A Compact Index Structure with High Data Retrieval Efficiency", International Conference on Service Systems and Service Management, pp. 1–5.

90. Jablonski, N. G. (2006). *Skin: A Natural History.* Berkeley: University of California Press.

91. Jati, H. & Selvam, D. D. D. P. (2008). Human skin detection using defined skin region. *In Proc. Information Technology, 2008. ITSim 2008. International Symposium on. 1.* doi: 10.1109/ITSIM.2008.4631637

92. Jia-Dong Ren, Jie Bao, and Hui-Yu Huang (2003). The Research On Spatio-Temporal Data Model And Related Data Mining. In *Proc. of International Conference on Machine Learning and Cybernetics*, pp 2-5

93. Jiawei-Han M. K. (2001) Data Mining: Concepts and Techniques, chapter 8, pages 335–393. Morgan Kaufmann Publishers

94. Jin, S., Kim, O., & Feng, W. (2013, June). MX-tree: a double hierarchical metric index with overlap reduction. In *International Conference on Computational Science and Its Applications* (pp. 574-589). Springer Berlin Heidelberg.

95. Joshi, D., Soh, L., Samal, A., & Zhang, J. (2014). A dissimilarity function for geospatial polygons. *Knowledge and Information Systems, 41*(1), 153-188. doi:10.1007/s10115-013-0666-2

96. Kakumanu, P., Makrogiannis, S., & Bourbakis, N. (2007). A survey of skin-colour modeling and detection methods. *Pattern recognition*, *40*(3), 1106-1122.

97. Kamel, Ibrahim, Faloutsos, Christos (1993): On Packing R-trees. In: Bhargava, Bharat K., Finin, Timothy W., Yesha, Yelena (eds.) CIKM 93 - Proceedings of the Second International Conference on Information and Knowledge Management November 1-5, 1993, Washington, DC, USA. pp. 490-499. http://doi.acm.org/10.1145/170088.170403

98. Kamlesh Kumar Pandey, Rajat Kumar Yadu, Anshu Dwivedi, Pradeep Kumar Shukla, (2015). "***A Analysis of Different Type of Advance database System For Data Mining Based on Basic Factor***", International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), ISSN: 2321-8169, PP: 456 - 460, DOI: 10.17762/ijritcc2321-8169.150206

99. Kang, S. Y., McGree, J., & Mengersen, K. (2015). Bayesian hierarchical models for analysing spatial point-based data at a grid level: A comparison of approaches. *Environmental and Ecological Statistics, 22*(2), 297-327. doi:10.1007/s10651-014-0299-y

100. Kang, C., Pugliese, A., Grant, J., & Subrahmanian, V. S. (2014). STUN: Querying spatio-temporal uncertain (social) networks. *Social Network Analysis and Mining, 4*(1), 1-19. doi:10.1007/s13278-014-0156-x

101. Kanagavalli, V. R., & Raja, K. (2013). A study on application of spatial data mining techniques for rural progress

102. Kaufman, L. and Rousseeuw, P. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley and Sons, 1990.

103. Kaur, J., Akanksha & Singh, H. (2018). Face detection and Recognition: A review. 6th International Conference on Advancements in Engineering & Technology (ICAET-2018), Feb. 23-24, 2018, Sangrur: ISBN No. 978-81-924893-3-9

104. Kaundinya, D. P., Balachandra, P., Ravindranath, N. H., & Ashok, V. (2013). A GIS (geographical information system)-based spatial data mining approach for optimal location and capacity planning of distributed biomass power generation facilities: A case study of tumkur district, india. *Energy, 52*, 77-88. doi:10.1016/j.energy.2013.02.011

105. Kawulok, M., Kawulok, J., & Nalepa, J. (2014). Spatial-based skin detection using discriminative skin-presence features. *Pattern Recognition Letters, 41*, 3-13.

106. Kemper, A., Grust, T., and Boehm, C. (2006). Advances in Database Technology -- EDBT 2006: 10 International Conference on Extending Database Technology, Munich, Germany, 26-31 March 2006, Proceedings

107. Kim, I., Shim, J. H., & Yang, J. (2003). Face detection. *Face Detection Project, EE368, Stanford University, 28*, 538.

108. Khan, R., Hanbury, A., & Stoettinger, J. (2010, September). Skin detection: A random forest approach. In *2010 IEEE International Conference on Image Processing* (pp. 4613-4616). IEEE.

109. Kohavi, R., Sommerfield D., Dougherty J., "Data Mining using MLC++, a Machine Learning Library in C++", International Journal of Artificial Intelligence Tools, Vol. 6, No. 4, pp. 537-566, 1997.

110. Kolkur, S., Kalbande, D., Shimpi, P., Bapat, C., & Jatakia, J. (2017). Human skin detection using RGB, HSV and YCbCr colour models. *arXiv preprint arXiv:1708.02694*.

111. Koperski K., Adhikary J., Han J (1996) "Knowledge Discovery in Spatial Databases: Progress and Challenges", Proc. SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, Technical Report 96-08, UBC, Vancouver, Canada,

112. Kim, S. K., Lee, J. H., Ryu, K. H., & Kim, U. (2014;2012;). A framework of spatial co-location pattern mining for ubiquitous GIS. *Multimedia Tools and Applications, 71*(1), 199-218. doi:10.1007/s11042-012-1007-2

113. Kriegel, H. P.; Kröger, P.; Zimek, A. (2009). "Clustering high-dimensional data". ACM Transactions Knowledge Discovery from Data 3: 1. doi:10.1145/1497577.1497578.

114. Kuan, J., & Lewis, P. (1997, September). Fast k nearest neighbour search for R-tree family. In Information, Communications and Signal Processing, 1997. ICICS., Proceedings of 1997 International Conference on (pp.924-928). IEEE.

115. Lakshmanan, V. (2012). *Automating the analysis of spatial grids: A practical guide to data mining geospatial images for human & environmental applications* (1;2012; ed.). Dordrecht: Springer New York. doi:10.1007/978-94-007-4075-4

116. Lazarevic, A., Fiez, T., & Obradovic, Z. (2000, January). A software system for spatial data analysis and modeling. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on* (pp. 10-pp). IEEE.

117. Lee, A. J. T., Chen, Y., & Ip, W. (2009). Mining frequent trajectory patterns in spatial–temporal databases. *Information Sciences, 179*(13), 2218-2231. doi:10.1016/j.ins.2009.02.016

118. Lee, Y. J., Lee, D. M., Ryu, S. J., & Chung, C. W. (1996, September). Controlled decomposition strategy for complex spatial objects. In *International Conference on Database and Expert Systems Applications* (pp. 207-223). Springer Berlin Heidelberg.

119. Legendre, P (1993). Spatial autocorrelation: Trouble or new paradigm? *Ecology*. 74, pp.1659–1673.

120. Leutenegger, S. T., Edgington, J. M. and M. A. Lopez. (1997) "STR: A simple and efficient algorithm," *in Proceedings 13th International Conference on Data Engineering*, p. 497–506

121. Li, D., Li, D. and Shuliang, W. (2015) Spatial Data Mining: Theory and Application. Springer-Verlag Berlin Heidelberg

122. Li, D. R., Wang. S. L. & Li, D, Y., (2006). Theory and Application of Spatial Data Mining (the first edition), Beijing, Science Press, 2006

123. Li, D., & Wang, S. (2005). Concepts, principles and applications of spatial data mining and knowledge discovery. In *Proceedings of the International Symposium on Spatio-Temporal Modeling,(STM'05), Beijing, China* (pp. 1-13).

124. Li, D., Wang, S., Yuan, H., & Li, D. (2016). Software and applications of spatial data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *6*(3), 84-114.

125. Li, Z., Ji, M., Lee, J. G., Tang, L. A., Yu, Y., Han, J., & Kays, R. (2010, June). Movemine: Mining moving object databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (pp. 1203-1206). ACM.

126. Lodi, A., Martello, S., and Monaci, M. (2002) "Two-dimensional packing problems: A survey". *European Journal of Operational Research*, 141: pp. 241–252. doi:10.1016/s0377- 2217(02)00123-6.

127. MacNab, Y. C., Read, S., Strong, M., Pearson, T., Maheswaran, R., & Goyder, E. (2014). Bayesian hierarchical modelling of noisy spatial rates on a modestly large and discontinuous irregular lattice. *Statistical Methods in Medical Research, 23*(6), 552-571. doi:10.1177/0962280214527386

128. Mamoulis, N. (2012) *Spatial data management* (1st ed.). US: Morgan & Claypool Publishers

129. Mansoori, E. G. (2014). GACH: A grid-based algorithm for hierarchical clustering of high-dimensional data. *Soft Computing, 18*(5), 905-922. doi:10.1007/s00500-013-1105-8

130. Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A. N., and Theodoridis, Y. (2010) *R-trees: Theory and Applications*. Springer Science and Business Media.

131. Mauder, M., Emrich, T., Kriegel, H., Renz, M., Trajcevski, G., & Züfle, A. (2015). Minimal spatio-temporal database repairs. Paper presented at the 492-495. doi:10.1145/2525314.2525468

132. Mazúr, E., & Urbánek, J. (1983). Space in geography. *GeoJournal*, *7*(2), 139-143.

133. Meng, X., Ding, Z., & Xu, J. (2014). *Moving objects management : Models, techniques and applications* (2;2nd 2014; ed.). Dordrecht: Springer Berlin Heidelberg. doi:10.1007/978-3-642-38276-5

134. Mokbel, M. F., Xiong, X. and Aref W. G., (2004) "Sina: Scalable incremental processing of continuous queries in spatio-temporal databases," in Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ACM, , pp. 623–634

135. Mortazavi T. M, Ebadati E. OM. (2019). *An improved human skin detection and localization by using machine-learning techniques in RGB and YCbCr colour spaces*. PeerJ reprints. https://doi.org/10.7287/peerj.preprints.27488v1

136. Moussalli, R., Absalyamov, I., Vieira, M. R., Najjar, W., & Tsotras, V. J. (2015;2014;). High performance FPGA and GPU complex pattern matching over spatio-temporal streams. *Geoinformatica, 19*(2), 405-434. doi:10.1007/s10707-014-0217-3

137. Nene, S. A., & Nayar, S. K. (1997). A simple algorithm for nearest neighbor search in high dimensions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(9), 989-1003.

138. Nguyen-Trang, T. (2018). A New Efficient Approach to Detect Skin in Colour Image Using Bayesian Classifier and Connected Component Algorithm. *Mathematical Problems in Engineering*. (*2018).* https://doi.org/10.1155/2018/5754604

139. Nishad, P. M. (2013). Various colour spaces and colour space conversion. *Journal of Global Research in Computer Science*, *4*(1), 44-48.

140. NIST, Information Technology Laboratory/Information Access. [Online] Available: https://www.nist.gov/itl/iad/imagegroup/colorferet-database [Accessed: 29th Mar-2019]

141. Noel Castree, Rob Kitchin, and Alisdair Rogers (2013) *spatial data mining* a dictionary of human geography (1st ed.) Oxford University Press.

142. Oliveira, V. A., & Conci, A. (2009). Skin Detection using HSV colour space. In *H. Pedrini, & J. Marques de Carvalho, Workshops of Sibgrapi* (pp. 1-2).

143. Omer Aftab, M., Javed, J., Bilal, M., Arfa H. & Adnan Khan, M. (2018). Implementation of NOGIE and NOWGIE for Human Skin Detection. *International Journal of Advanced Computer Science and Applications* (ijacsa), 9(7). http://dx.doi.org/10.14569/IJACSA.2018.090719.

144. O'Neill, B. (2006). *Elementary Differential Geometry (Rev. 2nd ed).* Academic Press: San Diego, CA.

145. Park, Y., Liu, L., and Yoo, J. (2013) "fast and compact indexing technique for moving objects" *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on* (pp. 562-569). IEEE.

146. Patel, P., and Garg, D. (2012) Comparison of Advance Tree Data Structures.arXiv preprint arXiv:1209.6495.

147. Patil, P. M. and Patil Y. M. (2012). Robust Skin Colour Detection and Tracking Algorithm. *International Journal of Engineering Research and Technology*. 1(8), ISSN: 2278- 0181.

148. Parsons, L., Haque, E., & Liu, H. (2004). Subspace clustering for high dimensional data: a review. ACM SIGKDD Explorations Newsletter, 6(1), 90-105

149. Patrick Oesterling, Patrick Jähnichen, Gerhard Heyer, and Gerik Scheuermann (2015). Topological visual analysis of clusterings in high-dimensional information spaces. it - Information Technology 1 57, 3-10, Walter de Gruyter GmbH, 10.1515/itit-2014-1073,

150. Peer, P. & Solina, F. (1999). An automatic human face detection method. Proc. Of Computer Vison Winter Workshop pp. 122-130.

151. Phung, S. L., Bouzerdoum, A., & Chai, D (2002). A novel skin colour model in ycbcr colour space and its application to human face detection. *In Proc. IEEE International Conference on Image Processing (ICIP' 2002).* ((1), pp. 289-292).

152. Preparata F. P. and Shamos M. I. (1985.) *Computational Geometry: An Introduction*. Springer-Verlag, New York.

153. Rao, F., Zhang, L., Yu, X. L., Li, Y., & Chen, Y. (2003, November). Spatial hierarchy and OLAP-favored search in spatial data warehouse. In Proceedings of the 6th ACM international workshop on Data warehousing and OLAP (pp. 48-55).

154. Ranjan, R., Patel, V. M., & Chellappa, R. (2017). Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(1), 121-135.

155. Ranjan, R., Bansal, A., Zheng, J., Xu, H., Gleason, J., Lu, B., & Chellappa, R. (2018). A fast and accurate system for face detection, identification, and verification. *arXiv preprint arXiv:1809.07586*. <Online at: https://arxiv.org/pdf/1809.07586.pdf: Accessed 19th March 2019>

156. Ramos, M. I., Cubillas, J. J., & Feito, F. R. (2015). Improvement of the prediction of drugs demand using spatial data mining tools. *Journal of Medical Systems*,*40*(1), 1-9. doi:10.1007/s10916-015-0379-

157. Ravada, S. (2014). Trends and Research Opportunities in Spatial Big Data Analytics and Cloud Computing: NCSU GeoSpatial Forum. [Online: https://cnr.ncsu.edu/geospatial/wp-content/uploads/sites/6/2016/04/Spatial-CloudRavada.pdf]

158. Jingbiao, R., & Shaohong, Y. (2010, July). Research and improvement of clustering algorithm in data mining. In *2010 2nd International Conference on Signal Processing Systems*(Vol. 1, pp. V1-842). IEEE.

159. Reza Najafi, M., & Moradkhani, H. (2013). Analysis of runoff extremes using spatial hierarchical bayesian modeling. *Water Resources Research*, *49*(10), 6656-6670. doi:10.1002/wrcr.20381

160. Romero, A. O. C. (2011). *Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach* (Doctoral dissertation, Master Thesis, University of Twente).

161. Rigaux, H. P., Scholl, M., & Voisard, A. (2003). Spatial Databases with Application to GIS. *SIGMOD Record*, *32*(4), 111.

162. Roheda, S. (2017). A multi-scale approach to skin pixel detection. *Electronic Imaging*, *2017*(4), 18-23.

163. Rossi, J. P. and Queneherv P. (1998). Relating species density to environmental variables in presence of spatial autocorrelation: A study case on soil nematodes distribution. *Ecography*. 21, pp.117–123.

164. Roussopoulos, N., & Leifker, D. (1985). Direct spatial search on pictorial databases using packed R-trees. *ACM Sigmod Record*, *14*(4), 17-31.

165. Roussopoulos, N., Kelley, S., & Vincent, F. (1995, June). Nearest neighbour queries. In ACM sigmod record (Vol. 24, No. 2, pp. 71-79). ACM.

166. Saberi, B., & Ghadiri, N. (2014). A Sample-Based Approach to Data Quality Assessment in Spatial Databases with Application to Mobile Trajectory Nearest-Neighbor Search. arXiv preprint arXiv:1409.2819.

167. Sagan, H. (1994) *Space-Filling Curve:*New York, Springer-Verlag.

168. Sagiroglu, S., & Sinanc, D. (2013, May). Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)* (pp. 42-47). IEEE.

169. Salman, H. A., Ibrahim, L. F., & Fayed, Z. (2013, January). Enhancing Clustering Technique to Plan Social Infrastructure Services. In 2013 4th International Conference on Intelligent Systems, Modelling and Simulation (pp. 18-23). IEEE.

170. Samet, H. (2006). Foundations of multidimensional and metric data structures. Morgan Kaufmann.

171. Samet, H. (1995). Spatial Data Structures Modern Database Systems: The Object Model, Interoperability, and Beyond: W. Kim, ed., Addison Wesley/ACM Press, Reading, MA, pp. 361-385.

172. Samet, H. (2009) "Sorting spatial data by spatial occupancy" *GeoSpatial Visual Analytics* (pp. 31-43). Springer Netherlands.

173. Samarah, S. (2012). Grid-based hierarchy structure for mining and querying vehicular ad-hoc networks. Paper presented at the 63-68. doi:10.1145/2386958.2386968.

174. Samson, G. (2012). *An Effective Approach for Mining Complex Spatial Dataset* (Masters' dissertation, University of Huddersfield).

175. Samson, G. L., Lu, J., Wang, L., & Wilson, D. (2013). An approach for mining complex spatial dataset. *Proceeding of Int'l Conference on Information and Knowledge Engineering*. Retrieved from http:// worldcompproceedings.com/proc/proc2013/ike/IKE_Papers.pdf

176. Samson**, G. L.,** Lu, J., and Showole, A. A. **(2014)** Mining Complex Spatial Patterns: Issues and Techniques. *Journal of Information & Knowledge Management.* 13 (02). *doi*: 10.1142/S0219649214500191.

177. Samson, G. L., & Lu, J. (2016). PaX-Dbscan: A Proposed Algorithm for Improved Clustering. In M. R. Pańkowska (Eds.) Studia Ekonomiczne. Zeszyty Naukowe, (269524th ed.). Katowice: Wydawnictwo Uniwersytetu Ekonomicznego w Katowicach. Retrieved from www.sbc.org.pl/Content/269524.

178. Samson, G. L., Lu, J., Usman, M. M., & Xu, Q. (2017). Spatial Databases: An Overview. In J. Lu, & Q. Xu (Eds.), Ontologies and Big Data Considerations for Effective Intelligence (pp. 111-149). Hershey, PA: IGI Global. doi:10.4018/978-1-5225-2058-0.ch003. Available at: http://www.igi-global.com/chapter/spatial-databases/177391.

179. **Samson, *G. L.,*** *Lu, J. and Xu, Q. (2016)* Large Spatial Datasets: Present Challenges, Future Opportunities. Int'l Conference on Change, Innovation, Informatics and Disrurptuive Technology, ICCIIDT London – UK, 2016. Available at: http://proceedings.sriweb.org/repository/index.php/ICCIIDT/icciidtt_london/paper/view/24

180. Samson, G. L., & Lu, J. (2018). Spatial Clustering in Large Databases Using Packed X-tree. *Egyptian Computer Science Journal*. ISSN: -1110-2586. 42(2). Pp.68-79.

181. Samson, G. L., Usman, M. M., Showole, A. A., Lu, J. & Jazzaa, H. (2018). Large Spatial Database Indexing with aX-tree. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, ISSN : 2456-3307, 3(3), pp.759-773.

182. Sander, J. (1998) Generalized Density-Based Clustering for Spatial Data Mining. München: Herbert Utz Verlag. ISBN 3-89675-469-6.

183. Sander, J., Ester, M., Kriegel, H and Xu, X (1998). "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications". Data Mining and Knowledge Discovery (Berlin: SpringerVerlag) 2 (2): pp169–194. Doi: 10.1023/A: 1009745219419.

184. Santos, M. Y., & Amaral, L. A. (2005). Geo-spatial data mining in the analysis of a demographic database. *Soft Computing, 9*(5), 374-384. doi:10.1007/s00500-004-0417-0

185. Schiller, J. (2004) "Location-Based Services" San Francisco, CA: Morgan Kaufmann, ISBN: 9781558609296

186. Schneider, M. (1999). Spatial Data Types: Conceptual Foundation for the Design and Implementation of Spatial Database Systems and GIS. In *Proceedings of 6th International Symposium on Spatial Databases*

187. Secchi, P., Vantini, S., and Vitelli, V. (2015) "Analysis of spatio-temporal mobile phone data: A case study in the metropolitan area of milan" *Statistical Methods & Applications, 24*(2), 279-300. doi:10.1007/s10260-014-0294-3

188. Shan, S., and Wang, G. G. (2010) "Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions" *Structural and Multidisciplinary Optimization, 41*(2), 219-241.

189. Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., Liu, X., & Lu, C. (1999). Spatial Databases - Accomplishments and Research Needs. *IEEE Transactions on Knowledge and Data Engineering*, *11*(1), 45–55. doi:10.1109/69.755614

190. Shekhar, S., Evans, M. R., Kang, J. M., & Mohan, P. (2011). Identifying patterns in spatial information: A survey of methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(3), 193-214.

191. Shekhar, S., Zhang, P. and Huang, Y. (2005) *Spatial Data Mining.* US: Springer. pp. 833-851.

192. Shirk, P. L., Linden, D. W., Patrick, D. A., Howell, K. M., Harper, E. B., Vonesh, J. R., & Fitzpatrick, M. (2014). Impact of habitat alteration on endemic afromontane chameleons: Evidence for historical population declines using hierarchical spatial modelling. *Diversity and Distributions*, *20*(10), 1186-1199. doi:10.1111/ddi.12239

193. Singh, S., & Prasad, S. V. A. V. (2018). Techniques and Challenges of Face Recognition: A Critical Review. *Procedia computer science*, *143*, 536-543.

194. Simonite, T. (2018). "Photo Algorithms ID White Men Fine—Black Women, Not So Much". *Wired*: https://www.wired.com/story/photo-lgorithms-id-white-men-fineblack-women-not-so-much. Accessed 13 March 2019.

195. Smit, A. J., Smit, J. M., Botterblom, G. J., & Mulder, D. J. (2013). Skin autofluorescence based decision tree in detection of impaired glucose tolerance and diabetes. *PLoS one*, *8*(6), e65592.

196. Soni, N. & Mate P. (2017). A Review on Face Detection and Recognization Techniques. International Journal of Scientific Research Engineering & Technology (IJSRET), ISSN: 2278 – 0882 Volume 6, Issue 1.

197. Song, Z., & Roussopoulos, N. (2001, July). K-nearest neighbor search for moving query point. In International Symposium on Spatial and Temporal Databases (pp. 79-96). Springer Berlin Heidelberg.

198. Steinbach, M., Ertöz, L., & Kumar, V. (2004). The challenges of clustering high dimensional data. In New directions in statistical physics (pp. 273-309). Springer Berlin Heidelberg.

199. Stuller, J., Pokorny, J., Bernhard, T. and Yoshifumi, M. (2000) Current Issues in Databases and Information Systems: East-European Conference on Advances in Databases and Information Systems Held Jointly with International Conference on Database Systems for Advanced Applications, ADBIS-DASFAA 2000 Prague, Czech Republic, September 5-9, 2000.

200. Sun, H-M. (2010). Skin detection for single images using dynamic skin colour modelling, *Pattern recognition*, 43 (4), pp. 1413-20.

201. Szczuka, M., Kryszkiewicz, M., Jensen, R. and Hu Q. eds. (2010) Rough Sets and Current Trends in Computing: Springer-verlag Berlin Heidelberg. Proceedings of the 7th International RSCTC Conference (2010), LNAI 6086, pp. 60- 69

202. Tan, W. R., Chan, C. S., Yogarajah, P., & Condell, J. (2012). A fusion approach for efficient human skin detection. *IEEE Transactions on Industrial Informatics*, *8*(1), 138-147.

203. Teegavarapu, R. S. V., Meskele, T., & Pathak, C. S. (2012;2011;). Geo-spatial grid-based transformations of precipitation estimates using spatial interpolation methods. *Computers and Geosciences*, *40*, 28-39. doi:10.1016/j.cageo.2011.07.004

204. Te-Hsiu Sun, Mingchih Chen, Shuchuan Lo, Fang-Chih Tien.(2007) "Face recognition using 2D and disparity eigen face", Expert Systems with Applications 33(2): 265-273

205. Thakkar, D (2018)."Top Five Biometrics: Face, Fingerprint, Iris, Palm and Voice". *Bayometric*: https://www.bayometric.com/biometrics-face-finger-iris-palm-voice. Accessed 13 March 2019.

206. Tian, Y., Ji, Y., & Scholer, J. (2015). A prototype spatio-temporal database built on top of relational database. Paper presented at the 14-19. doi:10.1109/ITNG.2015.8

207. Tikoo, S., & Malik, N. (2017). Detection, segmentation and recognition of face and its features using neural network. *arXiv preprint arXiv:1701.08259*.

208. Urbano, F., & Cagnacci, F. (2014). *Spatial database for GPS wildlife tracking data: A practical guide to creating a data management system with PostgreSQL/PostGIS and R* (1;2014; ed.). Cham: Springer. doi:10.1007/978-3-319-03743-1

209. UCI (1998). "Index of /ml/machine-learning-databases/iris". Online: Available at < https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data>

210. Velicanu, A., & Olaru, S. (2010). Optimizing Spatial Databases. *Informatica Economica*, *14*(2), 61–71.

211. Vezhnevets, V., Sazonov, V., & Andreeva, A. (2003, September). A survey on pixel-based skin colour detection techniques. In *Proc. Graphicon* ((3), pp. 85-92).

212. Vieira, M. R., Bakalov, P., & Tsotras, V. J. (2009). On-line discovery of flock patterns in spatio-temporal data. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems* (pp. 286-295). ACM.

213. Vieira, M. R. and Tsotras, V. J. (2013) *Spatio-Temporal Databases: Complex Motion Pattern Queries*. Springer,

214. Vijayalaksmi, S., & Punithavalli, M. (2012). A Fast Approach to Clustering Datasets using DBSCAN and Pruning Algorithms. International Journal of Computer Applications, 60(14).

215. Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, *1*, 511-518.

216. Wang, M., & Deng, W. (2018). Deep face recognition: a survey. *arXiv preprint arXiv:1804.06655*.

217. crime related factors—a spatial data mining approach. *Applied Intelligence, 39*(4), 772-781. doi:10.1007/s10489-012-0400-x

218. Wang, S., & Eick, C. F. (2014). A polygon-based clustering and analysis framework for mining spatial datasets. *Geoinformatica, 18*(3), 569-594. doi:10.1007/s10707-013-0190-2

219. Wang, D., Ding, W., Lo, H., Stepinski, T., Salazar, J., & Morabito, M. (2013). Crime hotspot mapping using the Zaniolo (eds.) Proceedings of 20th International Conference on Very Large Data Bases (VLDB'94), Santiago

220. Weaver, W. (1948) 'Science and complexity', *American Scientist.* 36, pg. 536-544.

221. Weaver, W. (1958) A quarter century in the natural sciences, Annual Report, New York: The Rockefeller Foundation

222. Welton, B., Samanas, E., and Miller, B. P. (2013, November). Mr. Scan: Extreme scale density-based clustering using a tree-based network of gpgpu nodes. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (p. 84). ACM.

223. Wilson, A. G. (2002) 'Complex Spatial Systems: Challenge for the Modeller', *Mathematical and Computer Modelling*. 36, pg 379 – 387

224. Wu, F., Zhan, J., Yan, H., Shi, C., & Huang, J. (2013). Land cover mapping based on multisource spatial data mining approach for climate simulation: A case study in the farming-pastoral ecotone of north china. *Advances in Meteorology, 2013*, 1-12. doi:10.1155/2013/520803

225. Xu, T., Wang, Y., & Zhang, Z. (2013). Pixel-wise skin colour detection based on flexible neural tree. IET Image Processing. 7(8), 751-761. Doi: 10.1049/iet-ipr.2012.0657

226. Xn hgvHeinecke, A., & Pflüger, D. (2013). Emerging architectures enable to boost massively parallel data mining using adaptive sparse grids. *International Journal of Parallel Programming*, *41*(3), 357-399. doi:10.1007/s10766-012-0202-0

227. Xu, X., J. Jager, and H.-P. Kriegel (1999) A Fast Parallel Clustering Algorithm for Large Spatial Databases. Data Mining and Knowledge Discovery, 3(3):263{290,

228. Yang, M. H., Kriegman, D. J., & Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, *24*(1), 34-58.

229. Yannis Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Mike Hatzopoulos, Klemens Boehm, Zafeiriou, S., Zhang, C., & Zhang, Z. (2015). A survey on face detection in the wild: past, present and future. *Computer Vision and Image Understanding*, *138*, 1-24.

230. Zhang, J., Wang, H., Davoine, F., & Pan, C. (2012, November). Skin detection via linear regression tree. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)* (pp. 1711-1714). IEEE.

231. Zortea, M., Flores, E., and Scharcanski, J. (2017). A simple weighted thresholding method for the segmentation of pigmented skin lesions in macroscopic images. *Pattern Recognition*, *64*, pp. 92-104.

232. Zhong, Y., Han, J., Zhang, T., Li, Z., Fang, J., & Chen, G. (2012, May). Towards parallel spatial query processing for big spatial data. In Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International (pp. 2085-2094). IEEE