


RESEARCH

Open Access



# Multi-dimensional geospatial data mining in a distributed environment using MapReduce

Mazin Alkathiri<sup>1\*</sup> , Abdul Jhummarwala<sup>2</sup> and M. B. Potdar<sup>2</sup>

\*Correspondence:

malkthere@gmail.com

<sup>1</sup> Administrative Sciences

College Hadhramout,

University of Science

and Technology,

Hadhramout, Yemen

Full list of author information

is available at the end of the article

## Abstract

Data mining and machine learning techniques for processing raster data consider a single spectral band of data at a time. The individual results are combined to obtain the final output. The essence of related multi-spectral information is lost when the bands are considered independently. The proposed platform is based on Apache Hadoop ecosystem and supports performing analysis on large amounts of multispectral raster data using MapReduce. A novel technique of transforming the spectral space to the geometrical space is also proposed. The technique allows to consider multiple bands coherently. The results of clustering  $10^6$  pixels for multiband imagery with widely used GIS software have been tested and other machine learning methods are planned to be incorporated in the platform. The platform is scalable to support tens of spectral bands. The results from our platform were found to be better and are also available faster due to application of distributed processing.

**Keywords:** Multiband raster processing, Multi-dimensional data processing, Geospatial processing, Spectral to geometrical space, K-means clustering

## Introduction

Satellites orbiting the Earth with their remote sensing capabilities captures information about the geography of Earth in form of remotely sensed images. These images are representations of the Earth's surface as seen from space and contains intensity about the physical quantities such as the solar radiance reflected from the ground, emitted infrared radiation or backscattered radar intensity [14]. This information is captured by multiple sensors on board the satellites which capture radiation for various wavelengths and is provided in the form of multispectral raster data. The use of multiple sensors for the same geographic area captures various types of information which includes thermal imaging (infrared), visible radiation (Blue, Green and Red), etc., and is stored as individual bands [2]. Multi spectral and multi-dimensional data is usually available in form of multi-band georeferenced tagged image file format (GeoTIFF) files, an extension of TIFF format. A Landsat 7 image comes in the form of a GeoTIFF file consisting of 8 spectral bands and each of the spectral band stores a different wavelength scattered or emitted from the Earth's surface. The earlier GeoTIFF standard was limited to supporting 4 GB of raster data which has been superseded by the current Big GeoTIFF standard and allows storage of image files larger than 4 GB in the TIFF container [17]. This was required due to the increasing spatial resolution and number of concurrent bands that

needed to be stored for a geographic area. There is also a large availability of images of Giga Pixel resolution ( $10^9$  pixels) images from domains such as bio-technology and forensics which are also stored in Big GeoTIFF format. Organizing and managing this kind of data is in itself a huge task and processing of it requires designing parallel and distributed systems which will allow for faster processing of terabytes of data and provide the results in a limited amount of time.

Raster image consists of representation of geographic objects in a two-dimensional scene and it is a two dimensional array of individual picture elements called pixels arranged in columns and rows [45]. Each pixel individually represents information about an area on the Earth's surface. The information about the area is represented by an intensity value and a location address in a two dimensional image. While the intensity value is represented by the measured reflectance, the location is represented by (longitude, latitude) value for a geo-referenced image [43]. A single pixel in a multiband image has several values depending on the number of sensors which captured information for that geographic location. The individual bands are usually used independently depending upon the geospatial analysis required and the intermediary outputs combined to form the final results. All of these bands when used in conjunction for geospatial analysis will provide more accurate representation about the phenomena on the Earth's surface. There are many techniques available to store and organize the multiband data (pixels) of an image in binary files such as band sequential (BSQ), band interleaved by pixel (BIP), and band interleaved by line (BIL). The BIL format stores the data of the first pixel from all the different bands in the first row, and the data of the second pixel from all the different bands in the second row and so on [20]. One example of such format is the sensor's data that comes from French satellite [also known as SPOT (Satellite Pour l'Observation de la Terre which translates to Satellite for observation of Earth) data] [64]. This study uses a custom input format which is similar to BIL to overcome some of the difficulties which are faced when processing such binary data formats in a MapReduce environment. There is a separate section ("[Geometrical space to spectral space \(preparation phase\)](#)") discussing the details and the data format required as input to the developed mining framework.

The paper has the following structure. A review of the advancements in Big Geospatial data mining has been presented in "[Related works](#)" section. The novel approach of converting multispectral data to geometrical space is discussed and developed in "[Proposed methodology](#)" section. "[Result and discussion](#)" section provides an analysis summary of the obtained results. Finally, "[Conclusion and future work](#)" section concludes the paper and provides directions for further research.

### **Related works**

Due to the requirements for various applications related to planning and decision making, the Landsat 7 program was launched in 1999 [29, 36]. The planning and decision making include landuse change analysis, environment conservation and impact assessment, wildlife habitat mapping, disaster management, urban sprawl analysis, agriculture and horticulture, natural resource management and monitoring, etc. The Landsat 7 program served to make a complete temporal archive of cloud free images of the Earth and is still active after the launch of its successor Landsat 8 in 2013 [15]. The Earth's

surface as depicted by true colour on widely used web mapping services like Google Maps/Earth, Bing Maps, Yahoo Maps, etc., is based on colour enhanced Landsat 7 satellite imagery. In addition to the satellite imagery, geospatial data is also being acquired by use of aircrafts, unmanned aerial vehicles (drones) and ground based operations such as land surveys.

### **Big-geospatial data**

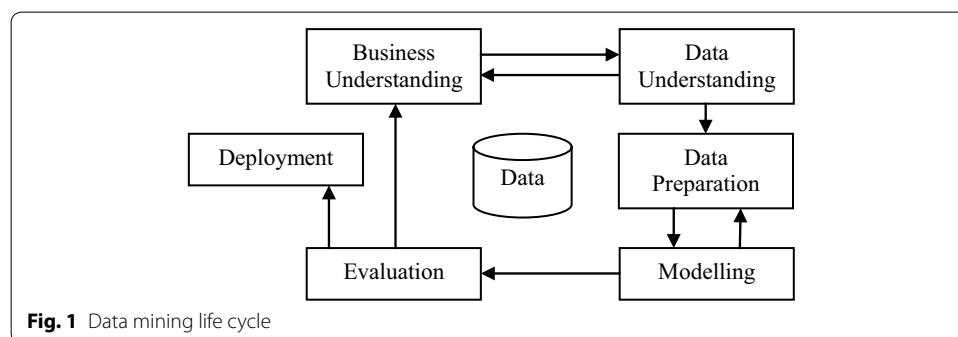
Collectively the geospatial data available from several sources has grown into petabytes and increases by terabytes in size every day [24]. The increase in the sources of data and its acquisition have been exponential as compared to the development of processing systems which can process the data in real time. Large amount of processed geospatial data is available for development of virtual globe applications from Nasa World Wind, temporal datasets archived at Google Earth Engine, etc. Beside these crowd sourcing online efforts such as OpenStreetMaps [5, 30] and Wikimapia have also assimilated terabytes of geospatial data. The data available from these efforts may have been derived from satellite imagery but is only applicable for a few applications such as for routing and navigation purposes. USGS [47], an organization established for the development of public maps and geo-sciences expertise has started providing access to applications and data related to disaster management during earthquakes, landslides, volcanoes, etc. but is limited in providing support for processing related to other planning and decision making applications. Private organizations such as earth observation system (EOS) have started providing automated on-the-fly earth observation (EO) imagery processing and analysis. Their products include providing realtime processing of classic GIS algorithms [21] on several of the open data sets available from the earth observation satellites (EOS).

The amount of geospatial data available is not just an increase in size but with availability of higher resolution has also increased the complexity of processing it and led to the geospatial “Big Data” phenomenon. According to Bhosale and Gadekar [8], the term ‘Big Data’ describes innovative techniques and technologies to capture, store, distribute, manage and analyze petabytes or larger-sized datasets with high-velocity and different structures. It is the data sets or combinations of data sets whose size (volume), complexity (variability), and rate of growth (velocity) make them difficult to be captured, managed, processed or analyzed by conventional technologies and tools. It has been stated for geospatial data in [48] that, “the size of spatial big data tends to exceeds the capacity of commonly used spatial computing systems owing to their volume, variety and velocity”, which truly encompasses the amount of spatial data available today and the complexity of the operations to be performed into the boundaries of the big data problem. The authors in [44] have supported that spatial data are large in quantity and are complex in structures and relationships. The study in [49] draws our attention to spatial interaction, spatial structure and spatial processes in addition to the spatial location which forms the basis of any spatial processing system.

The richness of information contained in raster data is only limited by the number of captured bands and its resolution. To derive the full benefits by processing such data it has become of utmost importance to overhaul existing multi-dimensional approaches and consider the geospatial characteristic of the data. This will not only ease and simplify the way geospatial data is processed and analyzed but will also allow

to further exploit the available richness of data. The variety in attributes that can be gathered from multiple spectrums for a geographic feature must be studied, visualized, interpreted and mined so as to extract qualitative, meaningful, useful information and new relationships. The results can provide insights into accurate geographic phenomena which is not available from analysis of individual bands. With the accumulation of large amount of data comes the difficult challenge of processing it and derive meaningful information which can be used for planning and decision making. The main aim of this work is to discover hidden knowledge from big geo-spatial data by considering multiple dimensions collectively for a geographical area rather than processing the bands individually. The novel approach of converting from spatial space to geometrical space preserves the essential multispectral characteristics of the data. The work addresses the shortcomings of existing approaches while processing big geospatial data and new distributed techniques required for processing both raster and vector data have been presented. In the present work, k-means clustering has been described in detail. The developed techniques can be adapted to several of the spatial data mining tasks including spatial prediction; spatial association rule mining; and temporal geo-visualization.

For processing raster data, image processing techniques have been well developed and are available with open source packages such as OpenCV, Scilab and other closed source packages and libraries. These are limited in scale and processing of giga-pixel scale images such as large multiband GeoTIFF files require tens of hours if not days. This inhibits the discovery of important knowledge, the realtime provision of which may be highly useful in applications of disaster relief, etc. Knowledge Discovery in Databases (KDD) is defined as the process of discovering useful knowledge from a collection of data and is closely related to data mining and it is important for the spatial data as well [35, 45]. The data mining process has been depicted in Fig. 1. It includes data preparation and selection, data cleansing, incorporating prior knowledge on data sets and interpreting accurate solutions from the observed results [63]. The data mining life cycle (DMMLC) starts with understanding the inputs or requirements, to formation of the system and until the last stage of deployment. Each of these depicted phases could be repeated in case the requirement changes. Geospatial data mining is an extension of classical data mining approach with the addition of geospatial component which requires application of complex image processing and spatial data processing techniques.



### Big-geospatial data processing

The classical data mining approach is no longer fit for processing of Big Data and has been modified and adapted by many frameworks which been developed to utilize the computing and storage available from distributed computing devices [37]. Big geo-spatial data adds another level of complexity to this Big Data ecosystem which now also requires considering the spatial and geographical location. This has furthered the complexity big data challenge [66]. A framework such as GS-Hadoop [31] can process millions of geospatial vector features in a matter of minutes but is limited to only processing vector data. De Smith et al. [19] have addressed the full spectrum of spatial analysis and associated modeling techniques that were available at the time with widely used geographic information systems (GIS) and associated softwares. The existing geospatial data processing systems are overwhelmed with the amount of data available and the complex operations required to be performed demands urgent development of tools capable of managing and analyzing such Big geospatial data [32]. Bradley et al. [12] aim to reduce the size of the data to be processed by identifying regions of the data that can be compressed, regions that must be kept, and regions that can be discarded. The transmission of high resolution raster images over low-bandwidth connections requires a great amount of time. This problem can be mitigated to a little extent by transmitting a series of low resolution approximations which converge to the final image [52]. Low bandwidth connections are no longer of concern due to the development of faster networks and internet bandwidth available to gigabit speeds for organizations [33, 54].

The above mentioned studies address a few of the shortcomings of traditional geoprocessing while some others [25, 68] extend the geoprocessing functionality to work upon parallel and distributed processing systems. Beside the complex processing of geospatial data, a considerable amount of work has been done on use of multidimensional data structures in information processing systems (IPS), the applications of which have been in fields of business analysis, astronomy, geomatics, bioinformatics, etc. [9]. The term “Multidimensional” essentially describes the way in which numerical information can be categorized and viewed [16]. It has been already established that large geospatial databases consist of multidimensional information. Several multidimensional models have also been proposed for establishing multidimensional databases (MDB) and on-line analytical processing (OLAP) [55]. It has also been stated that the traditional database systems are inappropriate for storage and analysis of multidimensional data since these systems are optimized for online transactional processing (OLTP) in which an enormous number of concurrent transactions containing normally few records are involved. Multidimensional data cannot be stored in OLTP databases. Geodatabases store multidimensional geospatial data with associated vector attributes and features for the raster data [6].

### Distributed processing of geospatial data

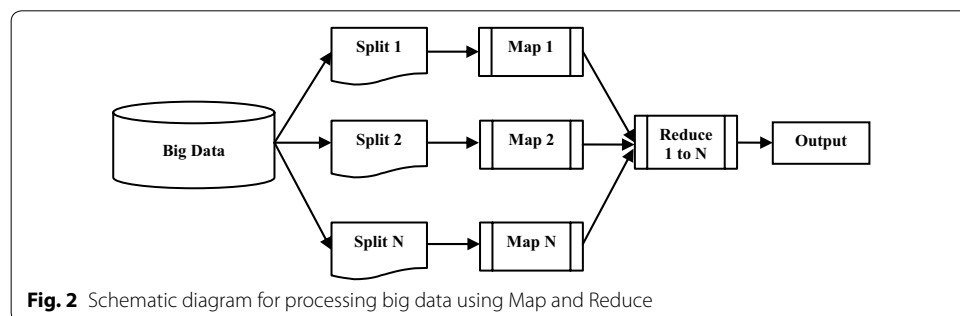
The distributed data processing framework MapReduce was first introduced by Google and later it was incorporated into Hadoop as its strong capability [46, 60]. Apache Hadoop is an open-source software for reliable, scalable, distributed computing on commodity hardware [27, 56]. Hadoop is one of the most widely used distributed processing

frameworks developed to address the challenges of big data. The framework is extensible and can be adapted to support big geospatial data. The main concept of the framework is segregated in two parts, viz., the Hadoop Distributed File System (HDFS) for storing data and the MapReduce programming model to process the data which is usually stored on HDFS.

The framework subsequently has been in development by the Apache Software Foundation. Apache defines Hadoop as software library framework that allows for the distributed processing of large datasets across clusters of computers using simple programming models [26]. It is important to highlight that initially Hadoop only supported MapReduce type of applications but has later been extended to support other programming paradigms [28]. Hadoop is capable of storing and managing large volumes of data efficiently, using a large number of commodity hardware as Nodes forming a Hadoop cluster. The same cluster is used for processing the data locally stored on the Nodes to reduce the network communication. Hadoop can be used for many applications involving large volume of Geospatial as well as Spatio-temporal data analysis, Biomedical Imagery analysis, simulation of various physical, chemical and computationally intensive application biological processes [2, 13, 18, 38].

MapReduce model of programming has become one of the best ways to processes big data which is inherently stored by Hadoop on its own distributed file system (HDFS) [10, 11]. The MapReduce model takes care of managing the whole processes from receiving the data, processing it and aggregating the results to form a single output. It takes care of distributing the data and managing the distributed resources throughout the whole processes. Applications which require to work with big data benefits hugely from HDFS as it provides high throughput access and streaming capabilities to large amounts of data. It has been developed to be fault-tolerant, can run on cheap commercially of the shelf (COTS) hardware and support streaming data. The main MapReduce phases include the Map and the Reduce phases which have been depicted in Fig. 2.

The Key-Value approach used by the Map phase groups input values with their associated keys. The keys along with their set of values will be sent to the Reduce phase and the required functions will be applied on those groups of values to get the needed output. There are other phases in MapReduce such as Shuffle phase, Sorting phase, Partitioner phase and Combine phase. The Combiner collects different (Key, Value) pairs, group similar keys and send them to the required node for the reducer. Keys and Values are sorted during the sorting phase. An appropriate partitioning logic can also be made available to ease the transferring of the data between the nodes.





To identify spatial patterns, most well-known statistical techniques are based on the concept of intra- and inter-cluster variances (like the k-means algorithm or the Empirical Orthogonal Function) [7]. There are various Classification and Clustering algorithms supported by Mahout, a data mining platform built on Hadoop. It supports k-means, canopy, fuzzy k-mean, naive bayes, etc. [26]. It should also be noted that these algorithms can be easily used with any framework based on Hadoop such as Apache Spark.

K-means algorithm is made to group a set of data into K sub-groups of the data or as we can say into K number of clusters, where the data can be in N dimensions, and in each cluster the sum of squares is minimized. Zhang et al. [67] improve the initial focal point and determine the K value, through simulation experiments while [1] propose new cost function and distance measure based on co-occurrence of values that works well for data with mixed numeric and categorical features. Sarode and Mishra [50] have mentioned, "It is not practical to require that the solution has minimal sum of squares against all partitions", except if the size of the data and dimensions is very small and the number of the clusters K is two.

Eldawy and Mokbel [23] developed SpatialHadoop, which is a comprehensive extension to Hadoop for support for geo-spatial vector data over Hadoop. It supports spatial constructs and the awareness of spatial data inside Hadoop code base. SpatialHadoop is composed of four main layers, which are language, storage, map-reduce and operations layer. The language layer provides Pigeon, a high level SQL-like language. The storage layer employs two level index structure of global and local indexing. And it introduces two components, spatialFileSplitter and spatialRecordReader, through the MapReduce layer. Finally the operation layer in reduce some basic spatial operations like range query, K-Nearest Neighbours (kNN), and spatial join, etc. SpatialHadoop is meant for the spatial data but it support only supports single dimension vector data and it does not have any of the data mining (classification and clustering) techniques listed above which may be required for processing satellite imagery [3].

Mennis and Guo [39] described the urgent need for effective and efficient methods to extract unknown and unexpected information from datasets of unprecedentedly large size having millions of observations, high dimensionality by hundreds of variables, and coming from heterogeneous data sources and having other complex attributes. Yao [65] stressed the development of spatial data infrastructure and efficient and effective spatio-temporal data mining methods. The development of CLARANS [42] is based on randomized search and is based on PAM and CLARA used for cluster analysis. Vatsavai et al. [58] studied into the IO and CPU requirements of spatial data mining algorithms for analyzing big spatial data and have presented the applications of bio-mass monitoring, complex object recognition, climate change studies, social media mining and mobility applications. STING [61] use hierarchical statistical information grid based approach for spatial data mining and STING+ [62] extends the approach by suspending the effects of the updates in the hierarchy until their cumulative effect triggers to multiple layers in the hierarchy. Bédard et al. [4] highlighted the requirement of efficient spatial data mining methods to cope with the huge size of spatial data which is increasing rapidly in the spatial data warehouses. To analyse the collected data at multiple resolutions, it is required to develop techniques with the ability to keep up the performance independent of the size of the spatial data.

A clustering model represented using choropleth to identify spatial relationships between the clustering obtained by spatial data mining has been developed using ArcView (a desktop GIS) and highlights the importance of correct visualization of geospatial data [41]. PixelMap [34] technique combines kernel-density-based clustering with visualization for displaying large amounts of spatial data. Visualization of big spatial datasets at various levels is an important requirement. The output from the proposed mining techniques can be scaled at various levels and passed to Desktop GIS for visualization.

### **Proposed methodology**

In this paper, the development and implementation of distributed framework for mining multiband raster geospatial data has been described. The framework has been evaluated using k-means clustering function which has also been updated to support our multi-dimensional data format in MapReduce environment. The proposed framework also supports multi-distance calculating functions such as the Euclidean distance and Manhattan distance while it is also simple to extend it to support other distance calculations such as Mahalanobis distance depending on the number of dimensions involved for data processing [51]. K-means clustering is a method commonly used to automatically partition a dataset into k-groups. It proceeds by selecting k initial cluster centers and then iteratively refining them as follows [59]:

- Each instance  $d_i$  is assigned to its closest cluster center.
- Each cluster center  $c_j$  is updated to be the mean of its constituent instances.
- The algorithm converges when there is no further change in assignment of instances to clusters.

In the present work, multi spectral (multi-dimensional) geospatial data derived from Landsat 8 have been used. To derive the experimental results, four to six spectral bands have been taken from several satellite images for the experimentations. The data is transformed for use with the developed mining platform. In the first stage, each pixel value of the different four bands are considered from the spectral space to the geometrical space. This has been further discussed in “[Geometrical space to spectral space \(preparation phase\)](#)” section of the paper. In the second stage, K-means clustering is applied in the MapReduce distributed mode and finally return the data into its initial form so it can be used for visualization. There are several implementations which have been based upon increasing the efficiency of k-means either supervised or non-supervised and there are several others which support multi-dimensional k-means clustering but none of them directly consider the information available in multispectral format. The present work can be easily extended for application of any other classification or clustering technique and any number of bands with simple modifications to support the proposed index file format. The proposed work forms one of the first distributed implementation for mining multi spectral data (supporting multiple dimensions) collectively and the techniques described can form a candidate for inclusion in the machine learning Mahout framework or for raster



processing support in other distributed geospatial processing systems. The performance of the mining platform has also been found to be satisfactory with respect to the amount of resources allocated and this has been highlighted in the succeeding sections.

**Geometrical space to spectral space (preparation phase)**

SpatialHadoop supports working with the geometrical location of the different features and implements spatial operations according to the type of shapes of the geo-spatial data which may be point, line, or polygon (rectangle). It does not support raster data. This study deals with the special case of working with multi spectral raster data available from Landsat 8 imagery in which every pixel can have 11 different values available from different bands. We use a subset of these bands All of these values are considered as the positional value of that pixel in different dimensions. In this way, all the different values of a single pixel are used to form a multidimensional spatial shape. E.g., polygon in a multi-dimensional space. The data in the geo-spatial mining process can then be used to perform the desired spatial operation. Table 1 describes the sample dataset.

Dataset description: predominantly limestone mining area  
 No. of bands: 6 (subset from Landsat 8 image has been taken)  
 Area: Aravalli Fort Hills (North of Gujarat, India)  
 Geographic Location:

Lat, Lon: 24° 00' North, 72° 54' East  
 Landsat 8 (path): (148,149); Landsat 8 (row): (43,44)

For implementation of distributed k-means for supporting multi-dimensional data, four bands from a raster image have been considered initially. The polygon thus formed with four points (one in each dimension) can be also reduced to a two dimension rectangle. Spatial operations can then be simplistically applied to this form of data. The following formula represents pixel values for four bands which have then been converted to a polygon and has been represented as a rectangle in two dimensions what is called as indexed pixel data and the process has been depicted in Fig. 3.

$$(X_1, Y_1, X_2, Y_2) \rightarrow \text{Polygon } (X_1, Y_1, X_2, Y_1, X_2, Y_2, X_1, Y_2, X_1, Y_1) \tag{1}$$

$$(45, 46, 47, 48) \rightarrow \text{Polygon } (45, 46, 47, 46, 47, 48, 45, 48, 45, 46) \tag{2}$$

**Table 1 Subset of the bands selected from the Landsat 8 image for experimentations**

Spatial band no.	Wavelength range (µm)	Description of band
2	0.450–0.515	Blue
3	0.525–0.600	Green
4	0.630–0.680	Red
5	0.845–0.885	Near IR
6	1.560–1.660	Short wave IR(SWIR)-1
7	2.100–2.300	Short wave IR(SWIR)-2

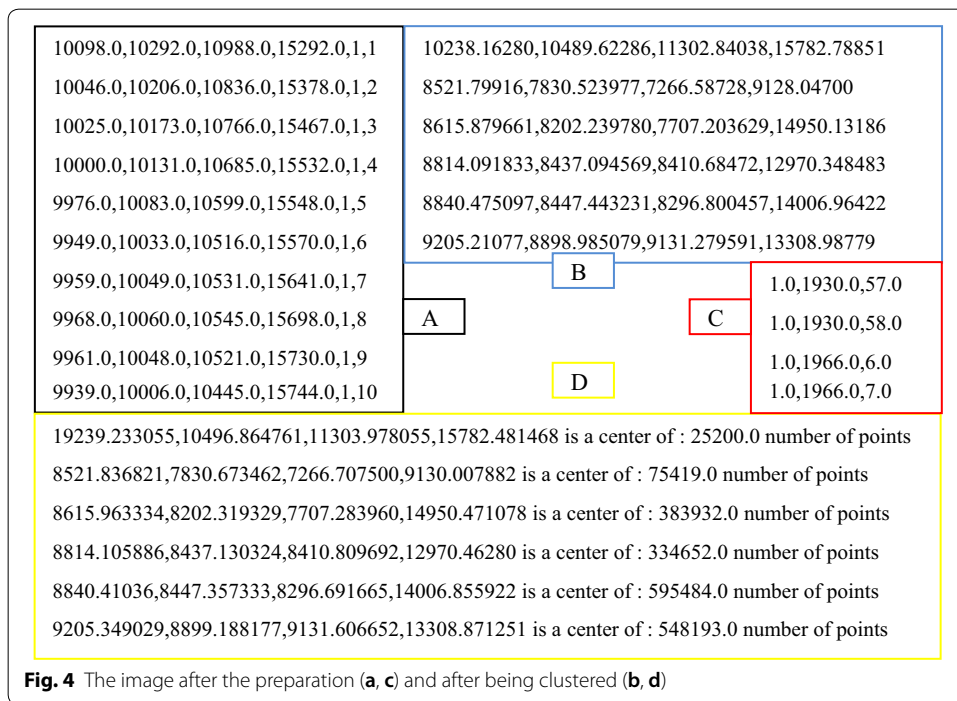
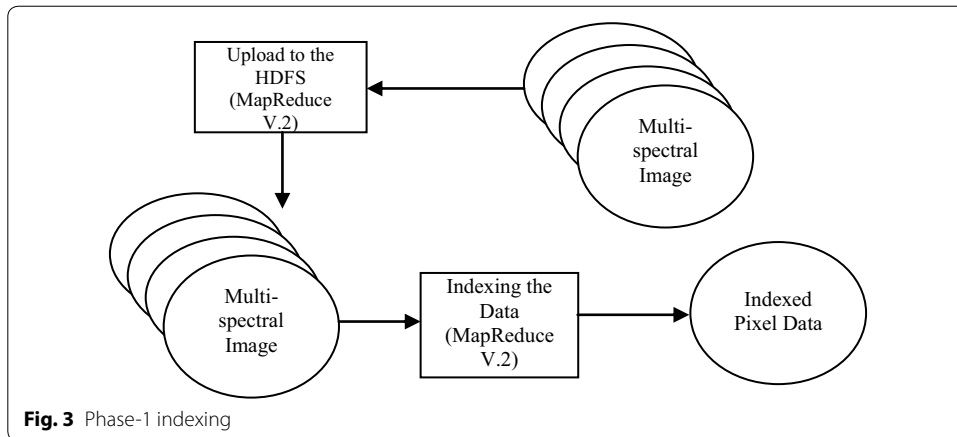
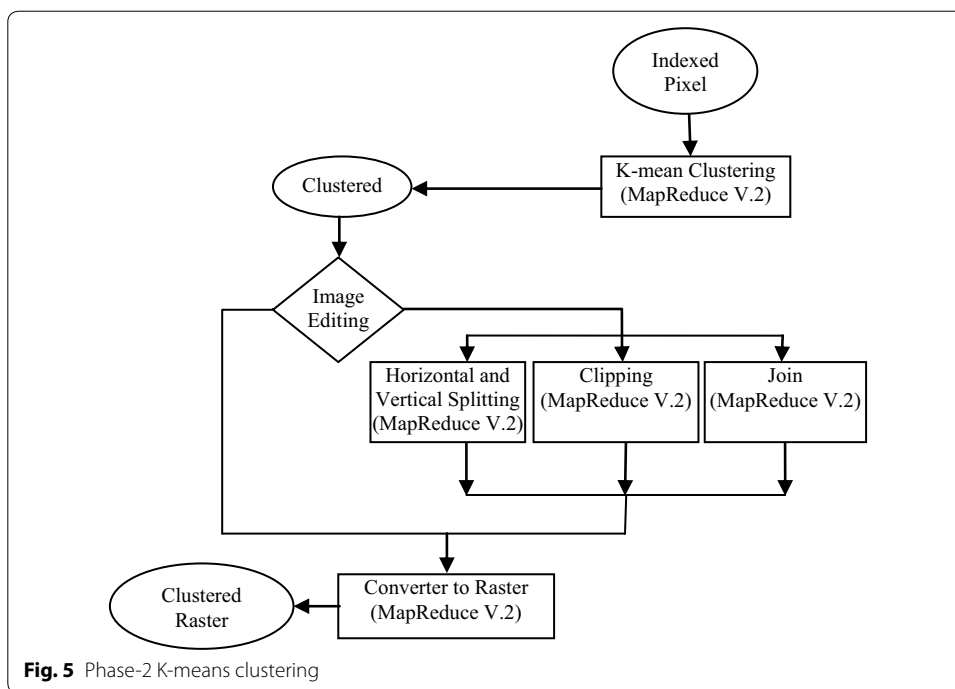


Figure 4 represents the total workflow which is divided into three main stages. In the first stage, the data is transformed it into 4-dimensional data set in which each pixel is transformed essentially into a rectangle owing to ‘4’ values obtained from each band of the image. The ‘4’ values for each pixel from every band is put into a resultant file which contains a matrix resembling the image’s pixels. The process can quickly iterate over thousands of rows and columns and in the resultant file each row contains the pixel’s values for the same geographic location from the ‘4’ different bands. These ‘4’ different values are represented as shown in Fig. 4a which also contains an index value unique to every pixel. The process is extensible to support ‘N’ number of bands.

The proposed mining mode demonstrates application of k-means clustering to work with multi-dimensional images in a MapReduce environment. Figure 5 represents the clustering and editing functions available for the input format. This work can further



be extended to support other geospatial operations available with SpatialHadoop or can alternatively be integrated with Mahout. The MapReduce implementation of the workflow and support for working with data stored on Hadoop Distributed File System (HDFS) opens opportunities for mainly supporting the big data ecosystem.

### Index all the pixels and assigning geographic location

#### A. Map Phase

The proposed k-means clustering model goes through the main two phases of the MapReduce programming module which are the Map and the Reduce phases [69]. In the Map phase, the k-means model will take two different files as the inputs, the data set file and the initial centroid file. It will calculate the distance from each point in the data input to each of the initial centroids. This way the nearest centroid to each point in the whole data set is obtained. The Map phase will then send to the reducer the values obtained for each point along with the nearest centroid to that point. The output from the Mapper to the Reducer task will include the nearest key (point) together with centroid values. Multi-spectral k-means clustering depicting transformation of values from Map to Reduce phases.

Input: A list of <key 1, value 1> pairs, k global centroids. Where value 1 is its content of online of the input file which contain the multi values of each pixel and its location.

Output: A list of <key 2, value 2> pairs. Where **key 2** is the index of the cluster and **value 2** is the point values and its location which belonging to that cluster (**key 2**).

## B. Reduce phase

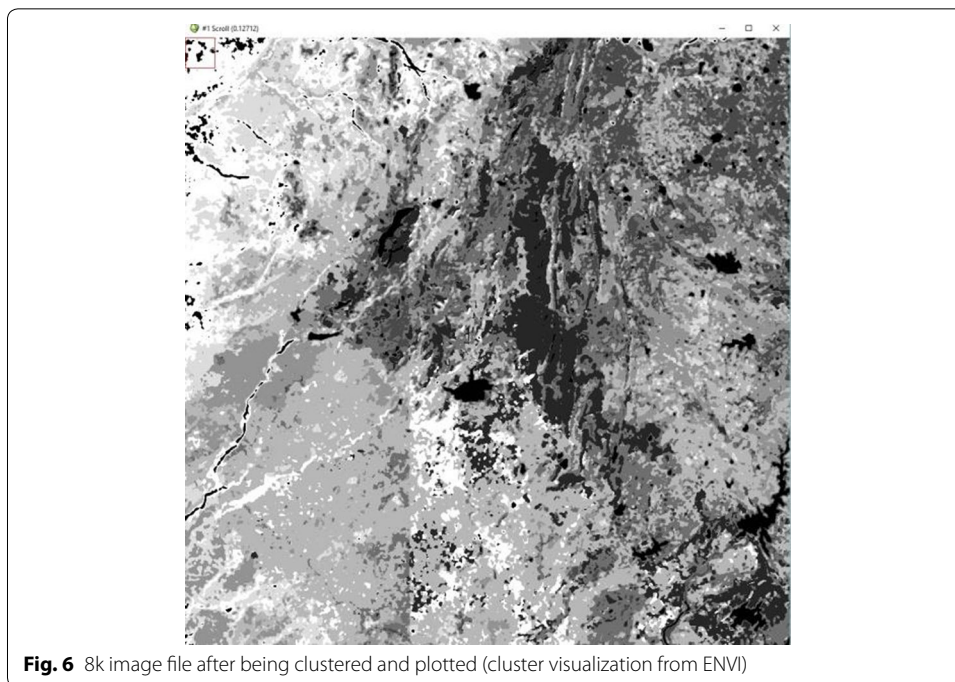
The reduce phase will receive each key with its attached group of values which are all the points from the data input for which the corresponding key is the nearest centroid. The main job of the Reducer is to calculate the optimal centroid out of each group of points which will have the average distant to all the element of that group of points. The reducer will produce the final output which is the new optimal centroids which again along the data input file will be taken to go through the Map and Reduce phases for the next iteration. The process is repeated till all the centroids get converged.

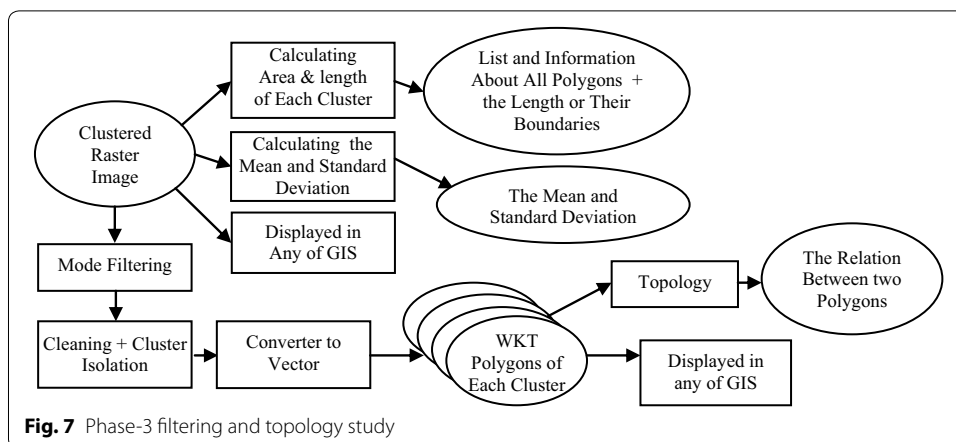
*Input: A list of <key 3, value 3> pairs Where key 3 is the index of the cluster and value 3 is the list of points values belonging to that cluster*

*Output: it will have two cases, one if the running iteration is any iteration but not the last, and the other case if the running iteration is the last iteration. And it can be described as:*

- *In the case of any iteration it will give the new calculated set of centroids.*
- *In case of the last iteration A list of <key 4, value 4> pairs will be added as an output. Where key 4 is the index of the cluster and value 3 is the position of the pixel in the image as shown in Fig. 4c*

The final output of the model will be two main files. The first file is the set of final centroids set, Fig. 4c. The second file will contain the coordinates of each point in the input dataset along with the cluster number which that point belong to as shown in Fig. 4d. Using the last output file from Reduce phase, we can get the clustered image back again for the visualization purposes, which is further done using MapReduce. Figure 6 represents the final clustering output from the MapReduce model. Figure 7 describes the phases for processing the image.





### Spectral space to geometrical space

The input multiband raster image is converted from geometrical space to the spectral space for processing purposes. Values from all the available bands from the image are considered. Those values represent different values of one pixel in an image and the same has been explained earlier. The location of that pixel is also added as the index to that same line where it has the pixel’s values similar to BIL format. Due to the present work, it has become possible to study and analyze multiband raster images without the need to process different bands individually and infer the phenomena for a particular area. After processing the image it is required to convert the obtained output from ASCII to image from the geometrical space to the spectral space, to be able to visualize the image or perform further processing and annotations that might be required after the mining process. The next part of the paper discusses several image processing functions that have been developed to work with multiband raster data on Hadoop.

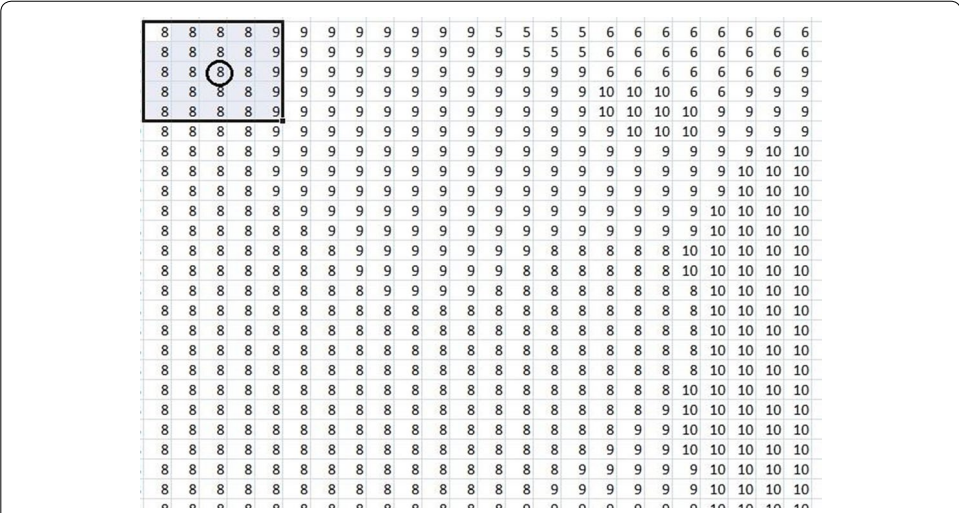
### Image filtering and after processing phase

#### Mode filtering

Mode filtering [57] model to the raster image, the mode filtering involves assigning to the central pixel the most common values inside the window around the pixel. Programs to work upon a distributed platform have been developed which will apply the mode filtering on the image. The window size that is needed by the algorithm can be specified at runtime. Mode filtering works to smoothen the edges of the polygons and at the same time to reduce the noise. Figure 8 shows the working of the filter. The window size here is (5 × 5) pixels, the filter will find out which value inside the window is the most common value and it will assign it to the widow’s central cell. The most common valued in the window, i.e., 8 will replace all the 9 values in the 5th column.

To execute the mode filtering, the following four arguments are required:

- <The input files >—the clustered image
- <File size >—the dimensions of the image
- <The window size>
- <The output file name>



**Fig. 8** Mode filtering process

In Fig. 9, a considerable amount of difference can be noticed after applying mode filter with different window size. The mode filtering can also be applied iteratively. An example image of the size 8000 × 8000 pixels is depicted in Fig. 9a. Figure 9b, c, d shows the results of mode filtering with window of the size 5 × 5, 9 × 9 and 11 × 11 pixels respectively on the same input image. One may select a window size appropriate for the data and to remove the desired amount of noise from the input image.

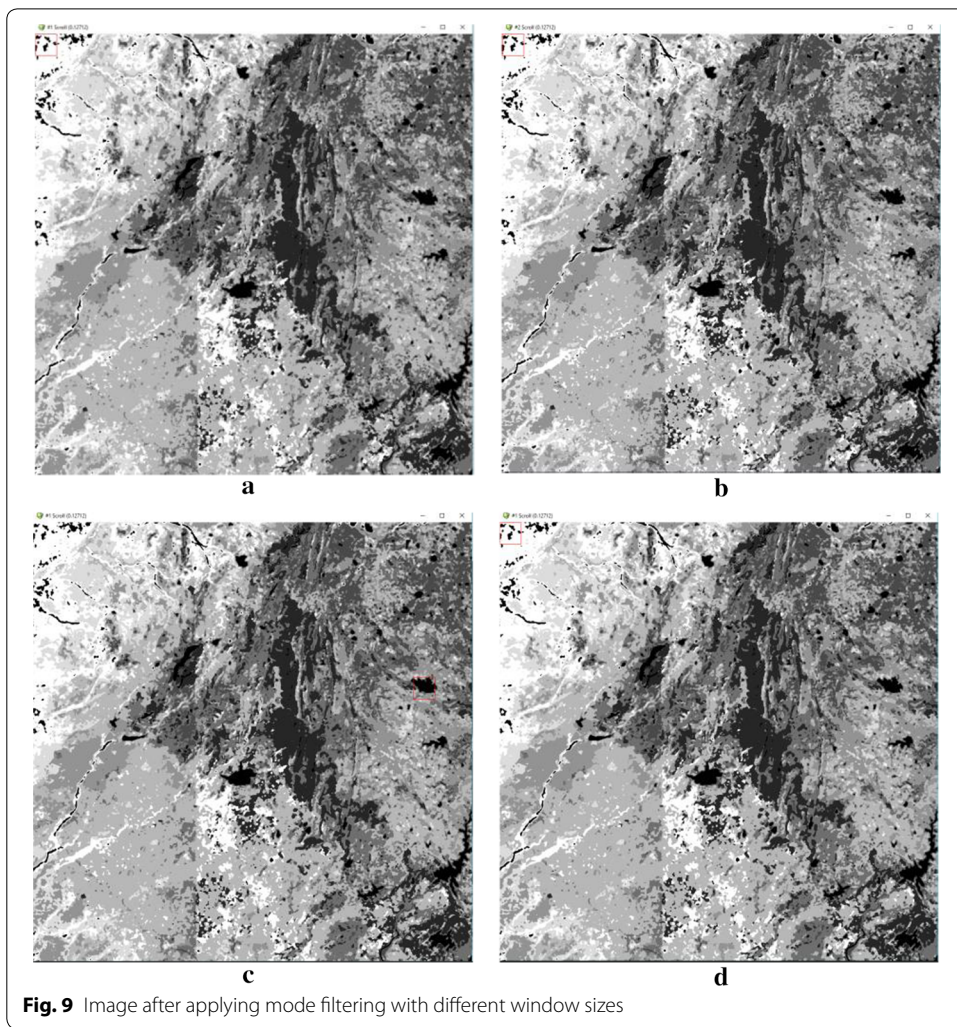
**Boundaries highlighting enhancement**

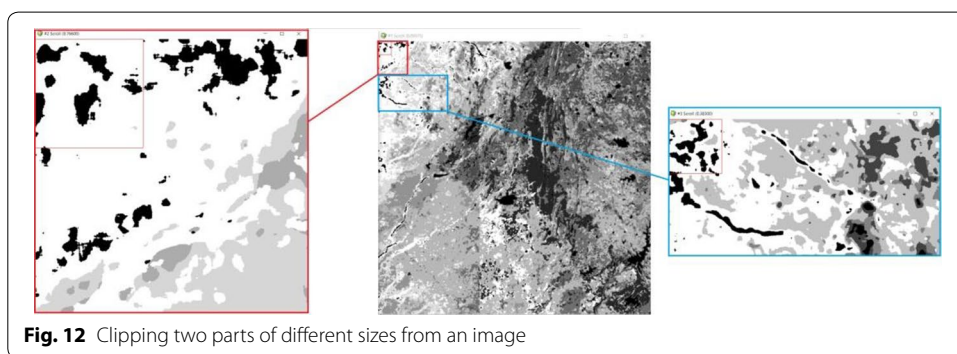
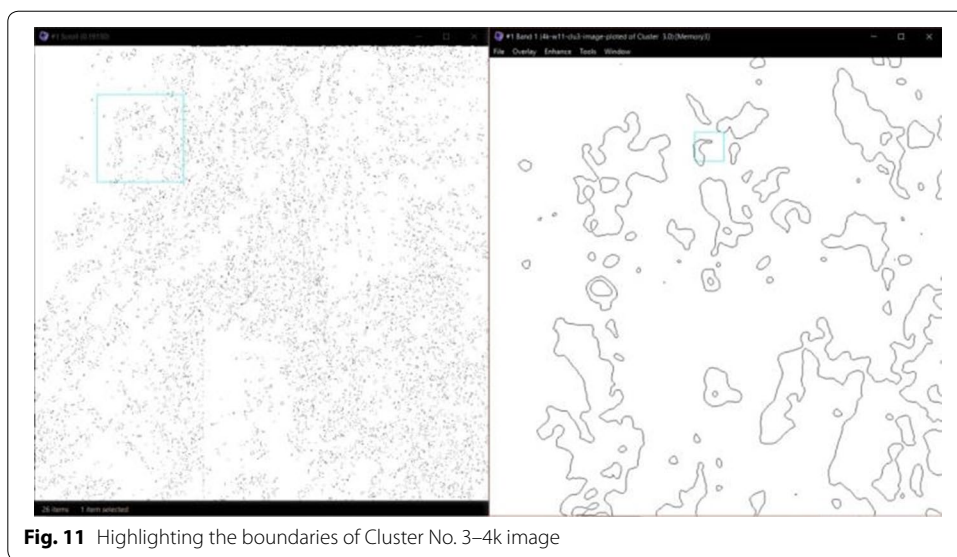
An application for vectorization has built to derive the boundaries of all the identified clusters so the output can be used with desktop GIS softwares for further analysis. Options to filter certain clusters or a group of clusters according to the requirements are available and can be specified as parameters when executing the vectorization tool. Two examples have been represented in Figs. 10 and 11. Figure 10 shows the polygons of Cluster No. 7 after filtering the results of the image of size of 8000 × 8000 pixels. Figure 11 shows the polygons of Cluster No. 3 filtered from the image of size 4000 × 4000 pixels. In both the figures, the left side contain the whole image and the right side contains a small part of the image (shown zoomed in). The visualization is accomplished using QGIS.

**Clipping**

Application for further editing including clipping any part of an image, splitting any image (horizontal/vertical) or for joining adjoining images. Figure 12 shows two different examples after clipping two different parts of an image. The left side is 1000 × 1000 pixels while a small part on the top left corner of the complete image is also highlighted with red boundaries. On the far right side of the another polygon with blue boundaries is clipped. The size of the clipped raster is 2000 × 1000 pixels.



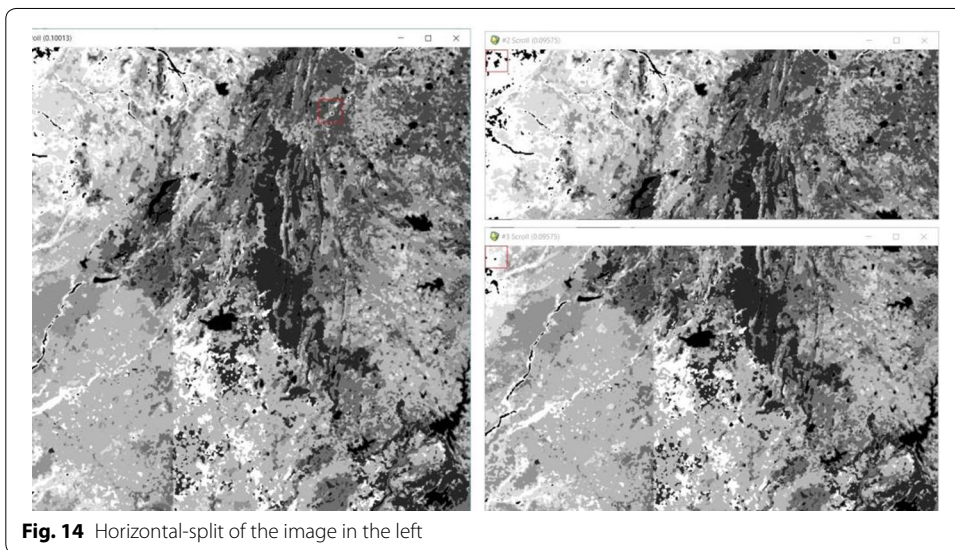
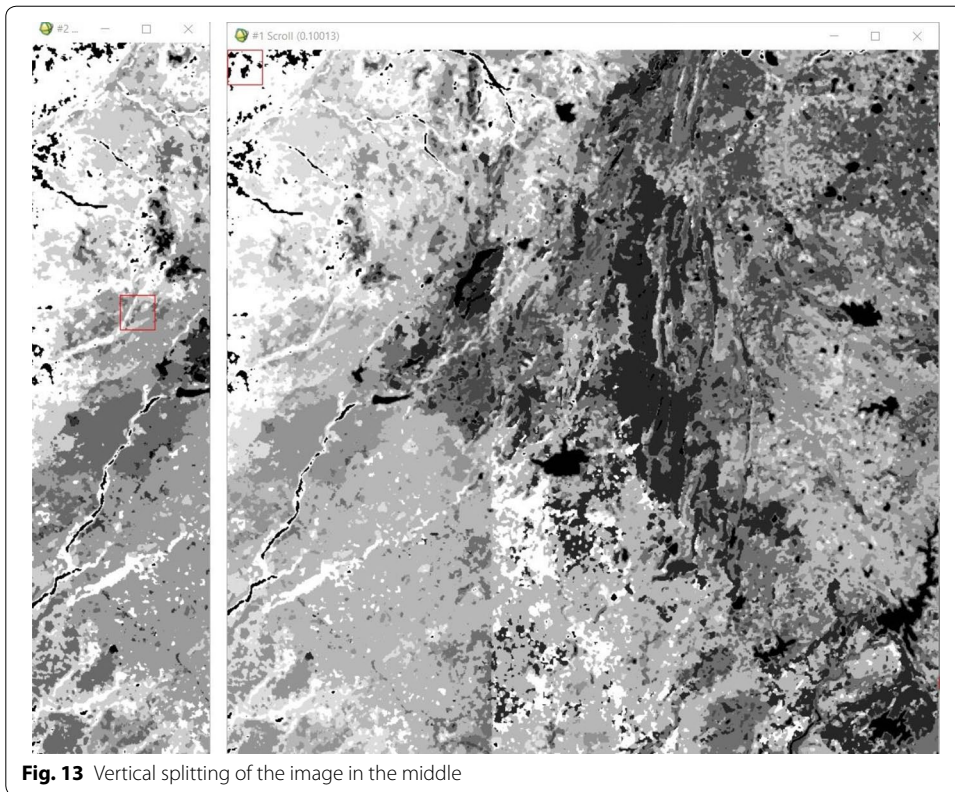




**Split**

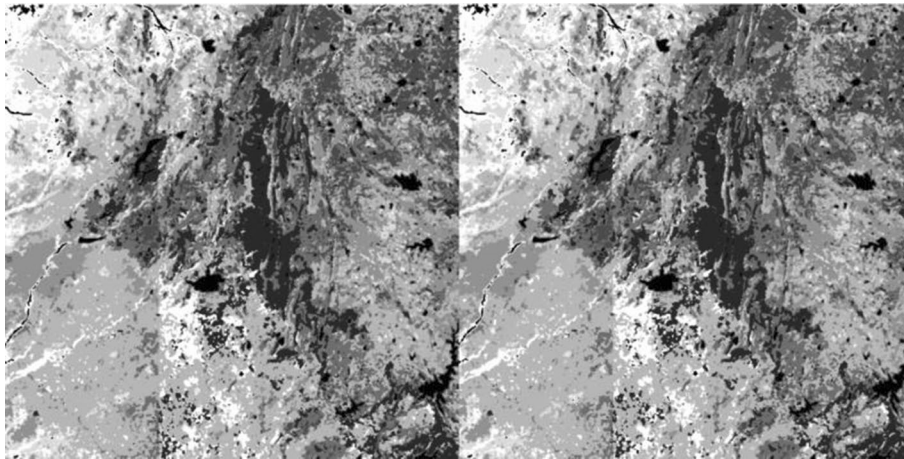
An editing tool for splitting images into multiple parts is also integrated with the data mining framework. This tool has two different modes; the first one is for vertical splitting and the second one is for horizontal splitting. After selecting the appropriate splitting method, the column(s) and/or row(s) for the split can be specified. The number of partitions or splits can also be specified and the application will automatically calculate the relevant rows and columns to be passed as arguments. Examples for both the cases have been presented. Figure 13 shows the vertical splitting in three different windows. The center window (highlighted in blue) shows the full image whereas the right and left windows shows the new images created after splitting the original image. Figure 14 depicts the same using horizontal splitting mode in which the left image represent the original image and on the right, the horizontally split parts are represented.





**Join**

Join is the editing tool that is made for joining any two images and it also have different modes, which are right, left, up, and down. Two input images and the mode for joining those two images is passed as an argument to attach the second images to the first one accordingly. This can be done to the images before the mining phase or after that



**Fig. 15** Joining two copies of the same image

according to the need. It has been made possible to process several images together after getting them from different sources, converting it to the proposed format and finally joining them together in a distributed environment. Figure 15 shows an example of an image that has been joined with itself using the Duplicate copy  $\rightarrow$  MapReduce join process.

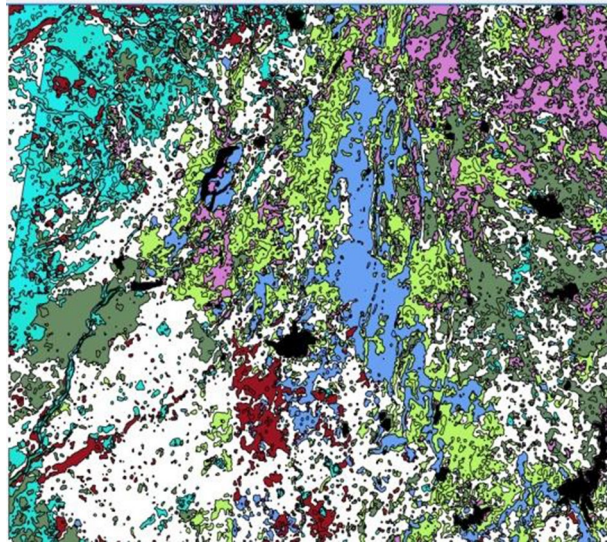
### **Cleaning**

Cleaning the image and removing small polygons (or clusters) is a technique called salt and pepper [53]. This technique has been modified to work in a distributed environment and has been integrated with the mining platform. To perform cleaning of the clustered image(s), a limit for the minimum size of polygons has to be specified which will remove all sub-clusters. Those small sub-clusters can be safely ignored while performing spatial operations or calculating statistics for large geographic areas. This function can also remove all the sub-clusters below a specified threshold and a clean output is obtained.

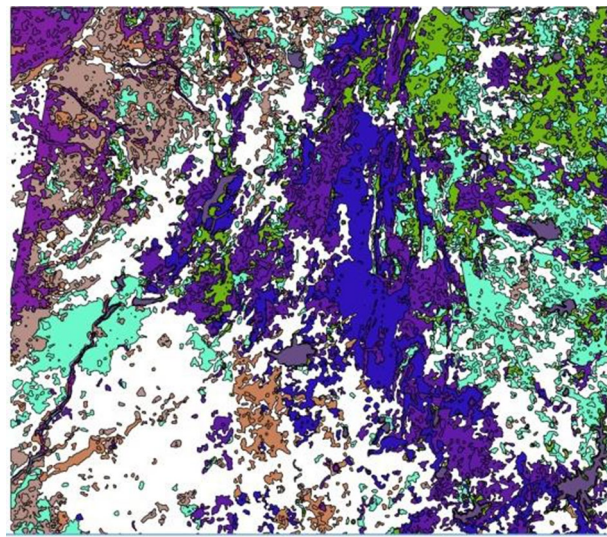
In Fig. 16, an image of  $8000 \times 8000$  pixels is represented after application of the above discussed techniques. Clustering have been applied and the noise in the resultant image is cleaned by application of mode filter with a window of size  $11 \times 11$  pixels. All the small sub-polygons with a threshold of the size  $100 \times 100$  or less have been cleaned. The resultant output is still left with several small polygons which may not be needed for further study. An appropriate threshold can also be decided automatically depending on certain parameters which can be specified by the user. A percentage of sub-clusters can be removed and which will only keep the required polygons which cover the maximum amount of geographical area.

Figure 17 represents results from the same image (of size of  $8000 \times 8000$  pixels) after application of the above discussed techniques. As discussed above, after clustering mode filtering is performed with a window size of  $11 \times 11$  pixels. The application automatically decides to clean all the small sub-clusters keeping the rest which covers more than 90% of the geographic area. The clustered data is filtered by automatically calculating





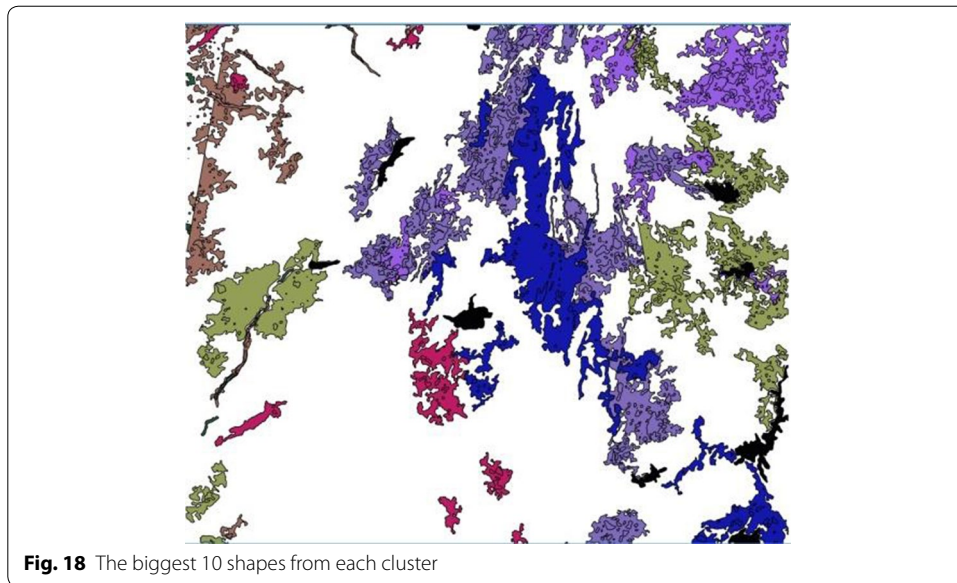
**Fig. 16** After removing shapes of size  $100 \times 100$  pixels and less



**Fig. 17** Removing the smallest 10% of shapes from each cluster

the minimum polygon size, which may be different for each cluster. E.g., in Cluster No. 1 (Blue), the threshold size of polygon which is not removed or cleaned is 1000 pixels. In Cluster No. 2 (Green) the minimum size of polygons is found to be 1694 and thus polygons which represent more than 90% of the area in each cluster will not be discarded.

Figure 18 represents output from the same input image. The application of the above discussed techniques is the same. For this sample, the biggest 10 sub-polygons from each cluster are kept where those are most likely to represent the area for further study. In each cluster, these top ten polygons represented a different percentage of the area for the cluster. E.g., in Cluster No. 4 (Purple), the top ten polygons



represent 48% of the cluster; The total area size and the top ten polygons from Cluster No. 10 (Blue) represent 77% of the area of the cluster.

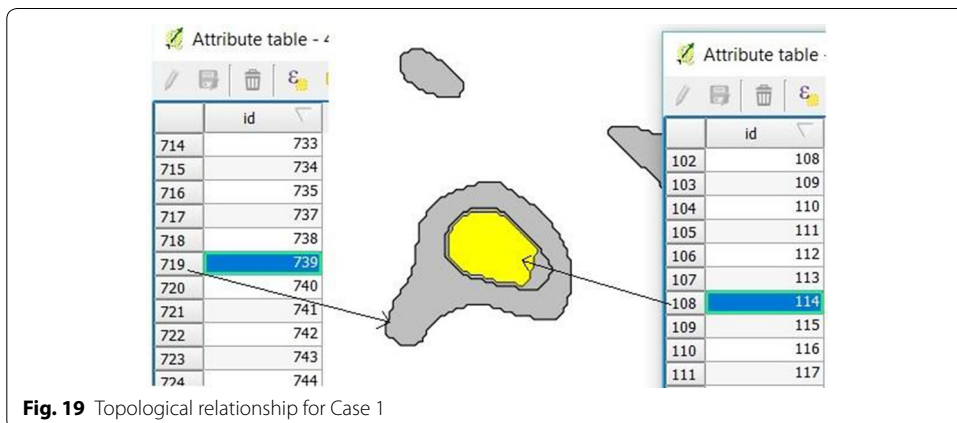
**Studying the polygons in the clusters**

The binary topological relation [22] between two objects A and B. is based on the intersection of the three part of each object which are the interior, boundary, and exterior of those two objects using the nine-intersection matrix which is shown below:

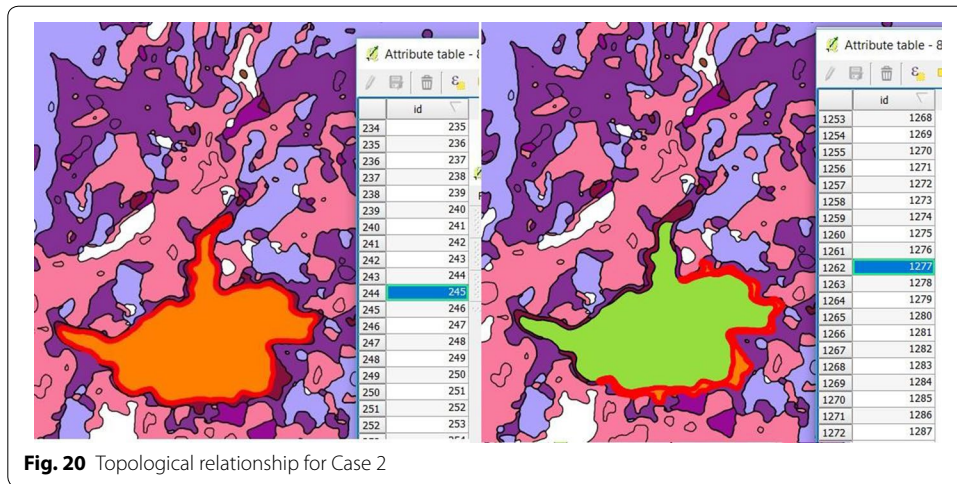
$$\Gamma_9(A, B) = \begin{pmatrix} A^o \cap B^o & A^o \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}. \tag{3}$$

where: *A*: object; *A<sup>o</sup>*: interior; *A<sup>-</sup>*: boundary and *∂A*: exterior, *B*: object; *B<sup>o</sup>*: interior; *B<sup>-</sup>*: boundary and *∂B*: exterior.

In Fig. 19, two polygons are taken from the 4 k data set after clustering. The first polygon is highlighted with a yellow colour and the other one is gray in colour. The







**Fig. 20** Topological relationship for Case 2

topological relationship is established by the application and it can be identified that polygon number 114 is inside polygon number 739 and they are not touching the boundaries. The binary topological relationship between these two objects is found with the following:

$$\text{Case 1: } \Gamma_9(114, 739) = \begin{pmatrix} 100 \\ 100 \\ 111 \end{pmatrix}. \tag{4}$$

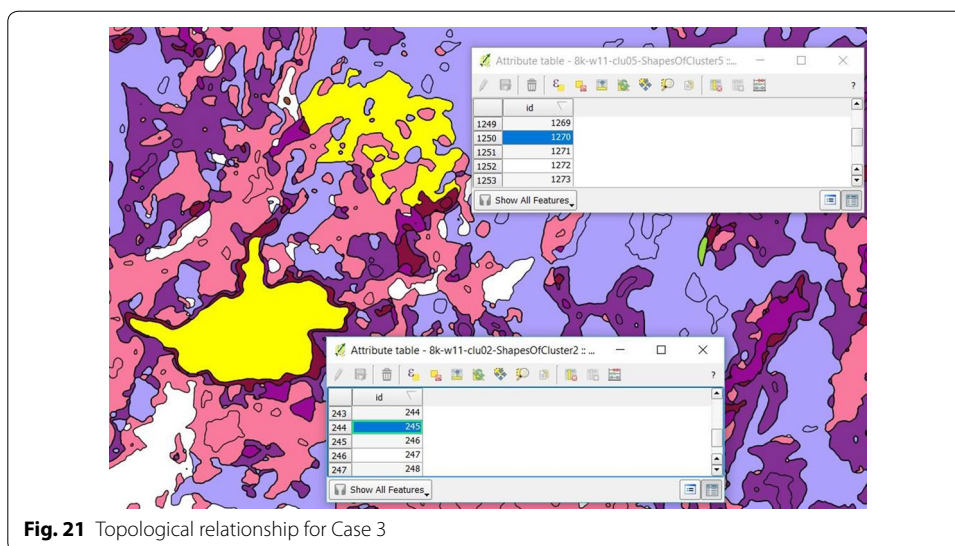
In Fig. 20, a polygon which is numbered 245 on the left side of the diagram belongs to the Cluster No. 2 that is outside of polygon number 1277 which belongs to Cluster No. 4. The polygon is shown on the right side of the diagram. Those polygons are touching each other's boundaries and we find the binary topological relationship between those two objects as the following:

$$\text{Case 2: } \Gamma_9(245, 1277) = \begin{pmatrix} 001 \\ 011 \\ 111 \end{pmatrix}. \tag{5}$$

In Fig. 21, two polygons have been selected and highlighted with a yellow colour. Polygon No. 245 is outside Polygon No. 1270 and they are not touching each other's boundaries. The binary topological relationship between those two objects is realized from the following:

$$\text{Case 3: } \Gamma_9(245, 1277) = \begin{pmatrix} 001 \\ 001 \\ 111 \end{pmatrix}. \tag{6}$$

An exhaustive list of relations between the largest polygons can be iteratively computed for topological study of the area. This list can then be further used to perform



**Fig. 21** Topological relationship for Case 3

statistical studies and application of other machine learning approaches to derive interactions between different environmental factors and conditions.

### Result and discussion

#### Technical specification of the Hadoop cluster used for the experiments

The Hadoop-cluster was set-up in a HP Proliant DL580 G7 server with 4 × Intel® Xeon® CPU E7-4870 @ 2.40 GHz totalling to 80 cores. The server is equipped with 512 GB of RAM. HPE 3PAR StoreServ served as storage backend with 2 × 8 Gbps connectivity. Hadoop (v.2.6.0) cluster was configured in the server with 50 Virtual Machines. The storage capacity of the cluster is 2.7 TB and which has been represented in Fig. 22. The cluster consisted of one Master machine (Name Node) and 50 data machines (Data Nodes). The Name Node is configured with 4 virtual processors and 12 GB RAM whereas the Data Nodes were heterogeneously configured with the following:

- 1 data node (on Name Node) with 4 virtual processor and 12 GB RAM
- 38 data nodes with 1 virtual processor and 8 GB RAM
- 11 data nodes with 1 virtual processor and 4 GB RAM

#### Data set preparation

The data used to test the functionality of the mining framework consists of two multi-spectral images of different size for the same location as described in “[Geometrical space to spectral space \(preparation phase\)](#)” section.

The pre-preparation phase: It consists of converting the multi-spectral data set into two dimension image by indexing all the pixels in the image obtained from different spectral bands using each pixel’s location. This generalization of multiple values for a single pixel obtained from multiple band into a single file also makes it easier for the data to processed irrespective of the number of bands from the image. As an example

```

hadoop@hadoop1:~$ hadoop dfsadmin -report live
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Configured Capacity: 2892315357184 (2.63 TB)
Present Capacity: 2421085233152 (2.20 TB)
DFS Remaining: 2379156168704 (2.16 TB)
DFS Used: 41929064448 (39.05 GB)
DFS Used%: 1.73%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0

-----
Live datanodes (50):

```

**Fig. 22** The storage capacity of the Hadoop cluster

the values of the first pixel taken from 6 different files (in case of 6 band image) are gathered in a single line along with the (number of the row and column which both represent the index for that pixel). The process is repeated for all the remaining pixels in the image. Everything is performed using MapReduce paradigm to utilize the potential of the Hadoop Cluster.

New format: From the pre-preparation phase, the example image with 6 bands is converted into two dimension image stored in an ASCII file which looks like the following:

```
#Hadoop fs -cat bigdata-256bs/part-r-00000 | head -n 10
```

```

8650,8075,7650,13779,11029,8582,1,10001
8653,8079,7659,13732,11065,8616,1,10007
8655,8082,7669,13701,11093,8642,1,10010
8660,8088,7680,13661,11131,8675,1,10016
8656,8079,7673,13603,11118,8675,1,10025
10275,10484,11249,17210,16363,13831,1,1003
8657,8076,7676,13537,11140,8699,1,10034
8658,8077,7667,13531,11058,8648,1,10043
8661,8080,7665,13518,10997,8615,1,10049
8657,8075,7655,13505,10958,8589,1,10052

```

The ASCII file is comma separated file and the first six values (in Red) represent the values of a pixel gathered from different bands whereas the last two values (in Black) represents the index of that pixel (geographic location). With this format it become easier to process the data for the pixel collectively using MapReduce.

### Benchmarking the Hadoop cluster (I/O)

As the storage backend consists of a single Storage Area Network (HPE 3PAR StorServ) with multiple disks, it is also important to test and benchmark the throughput of the HDFS. The benchmarking has been performed in conjunction with several suggestions provided by Mukherjee et al. [40]. It is appropriate to use a distributed file system such as HDFS on top of a shared disk infrastructure such as a storage area network. It is also possible to reduce the replication factor to 1 as redundancy and fault tolerance are not

**Table 2 Benchmarking the cluster**

TestDFSIO	No. of files	Total MBytes processed	Throughput MB/s	Average IO rate MB/s	IO rate std deviation	Test exec time (s)
Write	3	300	27.14932127	30.57173347	9.107456973	50.61
Read	3	300	45.91368228	46.32478333	4.366822343	58.072

desired in such an enterprise storage system. Table 2 shows various statistics from the DFSIO benchmark which ships with Hadoop.

### Running K-means clustering

In the above sections, several functions provided by the mining framework have been tested with many multiband images. The current section provides detailed discussion about the proposed MapReduce extensions to the k-means algorithm to work with multiband data in a geometric space. The results for clustering two multiband images for the same geographical location but with different resolutions using our approach to k-means have been described in detail. The proposed approach does not just perform clustering but with support for various image processing techniques allows to describe the geographical features in the image and in particular the topological relationships between different object that exist in the image. Those functions have already been described earlier.

The multiband images are uploaded to the HDFS and an indexed file is generated using the technique discussed in “Data set preparation” section using MapReduce. A parameter file containing the list of initial centroids is also provided for the uploaded image. In the subsequent testing, as the two different images are for the same geographic location with different resolutions, the same set of the initial centroids for clustering both the images is used. This will also help to compare the output from both of the images and to identify the quality of our clustering model. This has been further demonstrated in Table 7.

To test the performance of clustering, each image was clustered several times with a different block-size. The number of blocks that an image will be divided into depends on the size of the image and the block size that has been specified in the cluster configuration. A small block size will lead to a large number of blocks even for a small image while a large block size is configured if it is desired that the image is not split into a large number of block. Small block size is desired in case of small images so that several block are distributed across the cluster and thus cluster storage and computing capacity can be used. A large block size would reduce the network communication as several small block need not be communicated across the cluster of 50 nodes. The results of the clustering approach has been tested twice; once in a full Hadoop cluster consisting of 50 nodes and the second time with just two nodes.

The performance of the approach to cluster the multi-spectral raster images in the Hadoop framework (a distributed environment) and related image processing techniques is represented in Tables 3, 4, 5 and 6. Each of the tables shows the elapsed time

**Table 3 Clustering (1.9 GB) raster image with a 50 node Hadoop cluster**

Block size (MB)	Elapsed time (min)	Average map time (min)	Average shuffle time (min)	Average merge time (min)	Average reduce time (min)	Total mapping time (min)
16	4.7	2.4	2.0	0.0	0.1	3.3
32	4.1	1.1	1.3	0.0	0.3	2.8
64	4.5	1.8	1.0	0.0	0.3	2.8
96	5.8	2.4	2.9	0.0	0.1	4.5
128	6.4	3.2	4.6	0.1	0.1	5.3
256	8.0	4.3	6.1	0.0	0.1	6.8

**Table 4 Clustering (1.9 GB) raster image with a 2 node Hadoop cluster**

Block size (MB)	Elapsed time (min)	Average map time (min)	Average shuffle time (min)	Average merge time (min)	Average reduce time (min)	Total mapping time (min)
16	16.3	13.1	2.3	0.1	0.2	12.9
32	13.0	10.0	2.1	0.1	0.2	11.6
64	11.5	9.3	4.3	0.2	0.1	10.5
96	11.8	9.7	6.6	0.2	0.2	10.9
128	14.1	11.8	12.2	0.2	0.1	10.8
256	14.9	11.6	13.5	0.1	2.9	14.2

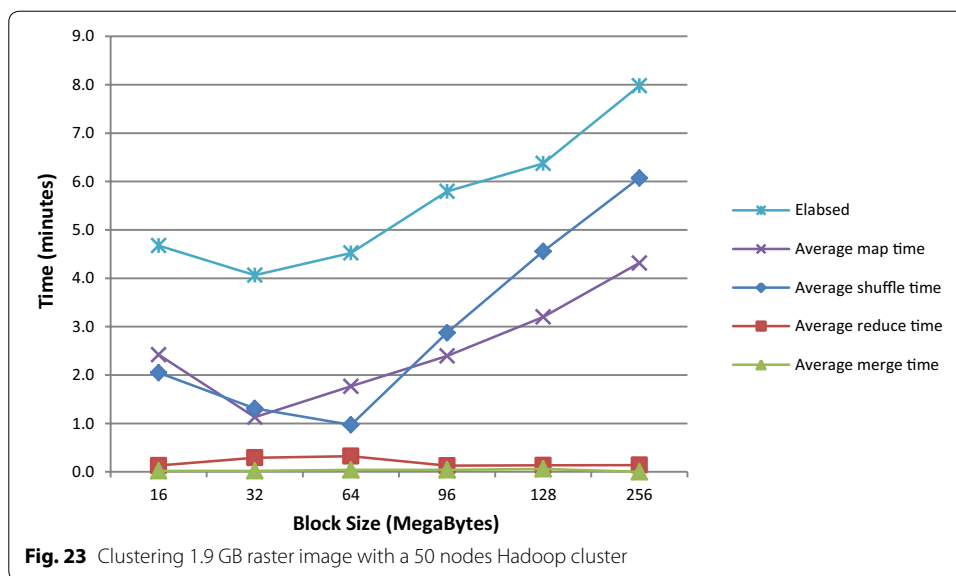
**Table 5 Clustering (17.79 GB) raster image using a 50 nodes Hadoop-cluster**

Block size (MB)	Elapsed time (min)	Average map time (min)	Average shuffle time (min)	Average merge time (min)	Average reduce time (min)	Total mapping time (min)
128	36.0	13.7	17.9	0.3	1.6	22.9
265	34.5	13.0	14.1	0.3	2.0	21.4
512	36.4	13.7	14.8	0.2	1.7	23.9
1024	49.5	23.5	32.7	0.5	1.9	34.6

**Table 6 Clustering 17.79 GB raster image using a 2 node Hadoop cluster**

Block size (MB)	Elapsed time (min)	Average map time (min)	Average shuffle time (min)	Average merge time (min)	Average reduce time (min)	Total mapping time (min)
128	99.472	61.95	22.262	0.274	1.96	88.426
265	85.7	69.7	14.1	0.3	1.5	74.2
512	74.0	58.6	10.3	0.3	1.8	63.9
1024	114.2	95.3	103.3	0.3	1.7	104.1
18 GB	146.3	114.1	136.5	0.0	1.3	137.0

of the k-means clustering process, the average time used for each of mapping, shuffling, merging, and reduce phases in addition to the total mapping time with respect to the block size.



**K-means using 50 node Hadoop cluster for Image No. 1**

Table 3 shows the several statistics of clustering the Image No. 1 with 8000 × 6000 pixels using a full Hadoop cluster of 50 nodes. It is evident that the minimum elapsed time was recorded in the time where the image was stored using the block size 32 MB, that was followed by the block size of 64 MB and so on till the maximum block size of 256 MB. The same is clearly illustrated in Fig. 23 and it can be seen that all of average mapping time, average shuffling time, and the totally mapping time increase with the block size. The average merge and reduce time was not affected with the change in the block size of the image.

Figure 23 clearly shows the increase in the execution time with respect to the block size.

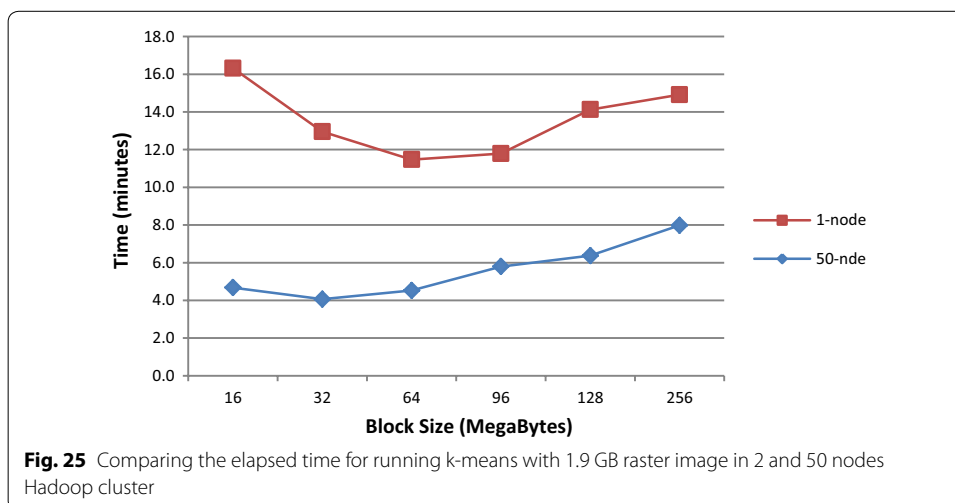
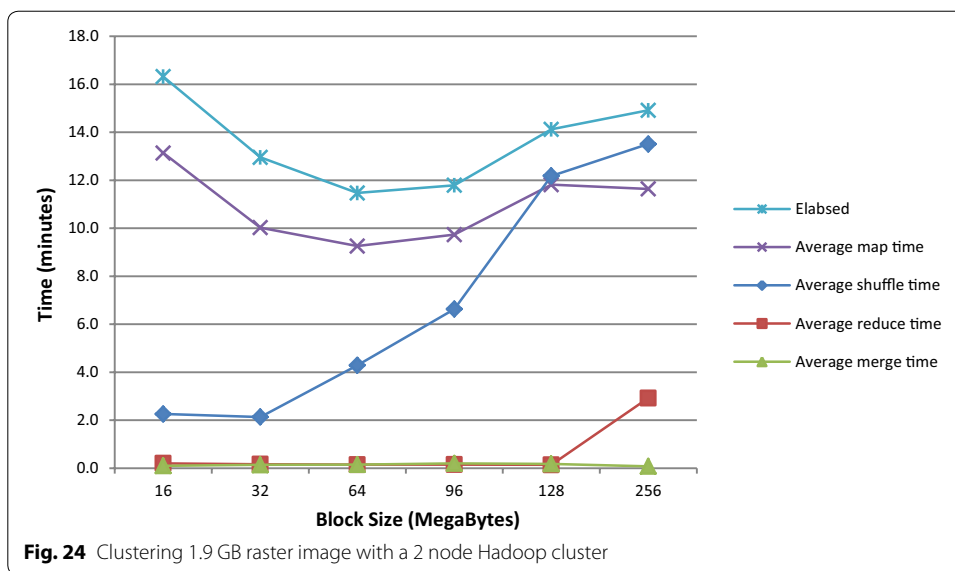
From the statistics we can infer that by increasing the block size, the number of the blocks created from the image will be reduced and due to less number of blocks, only a few number of Hadoop nodes (processing machines) will compute over the data. This is because several Data Nodes will not be even utilized due to non-availability of data local to them and this leads to increase in the execution time.

**K-means using 2 node Hadoop cluster for Image No. 1**

The Hadoop cluster was resized keeping only a couple of Data Nodes. Table 4 represents the average mapping, shuffling, merging, and reduce phases in addition to the total mapping time with respect to the block size. It is found that the execution time was the least in the case of the blocks with size of 64 MB. It was followed by 96, 128 and 256 MB block size respectively. It can be noticed from Fig. 24 that the execution time increases with the increase in the block size.

From the statistics of executing K-means clustering model over Hadoop clusters with 50 nodes, as available in Table 3, it is evident that the fastest execution time is 4.1 min with block size of 32 MB. From the statistics, available in Table 4, it can be noticed that the execution time has increase by 3 times due to limited number of nodes (= 2) and the



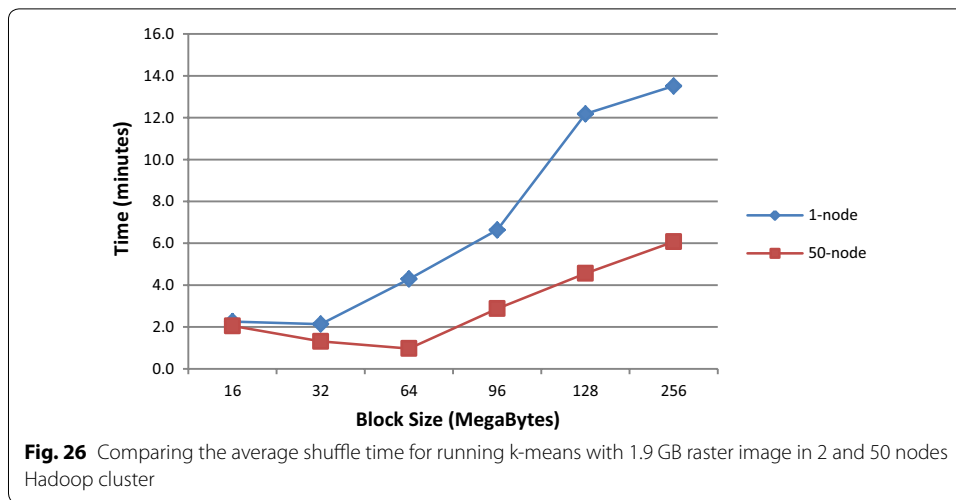


minimum execution time is 11.5 min with block size of 64 MB. Figure 25 illustrates the elapsed time for both the clusters varying the block size.

**Comparing results of K-means between 2 nodes and 50 nodes Hadoop cluster for Image No. 1**

Figure 26 shows a large increase in the time needed for shuffling the blocks with increasing the size of the blocks. The shuffle time remains considerably the same with up to 64 MB of block size for both 2 nodes and 50 nodes Hadoop cluster. For the 50 nodes cluster, the shuffle time varies from 1 min to 6.1 min for 64 MB to 256 MB block size respectively. For the 2 Nodes Hadoop cluster, the average shuffling time goes up to 13.5 min in case of 256 MB.

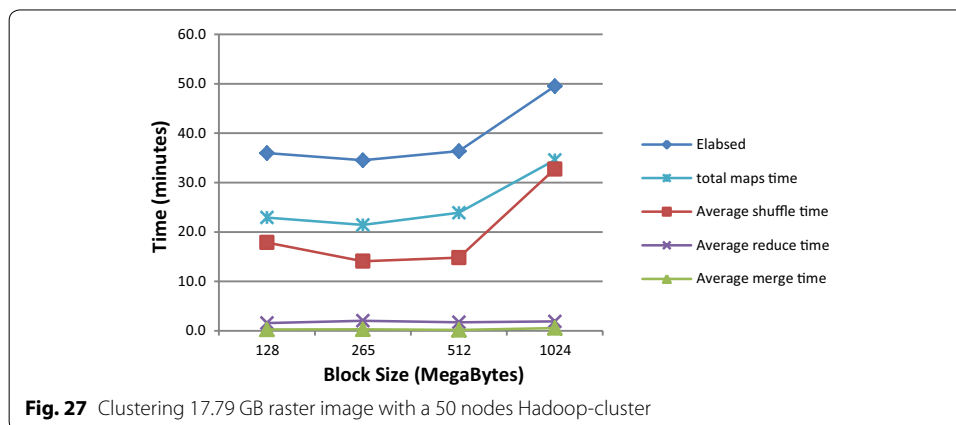
It has been mentioned before that the processing time increases in the case of bigger blocks of data. With bigger blocks there will be less number of blocks. E.g., if an image is of 1 GB size, with a block size of 256 MB, it will result in only 4 blocks. This means that

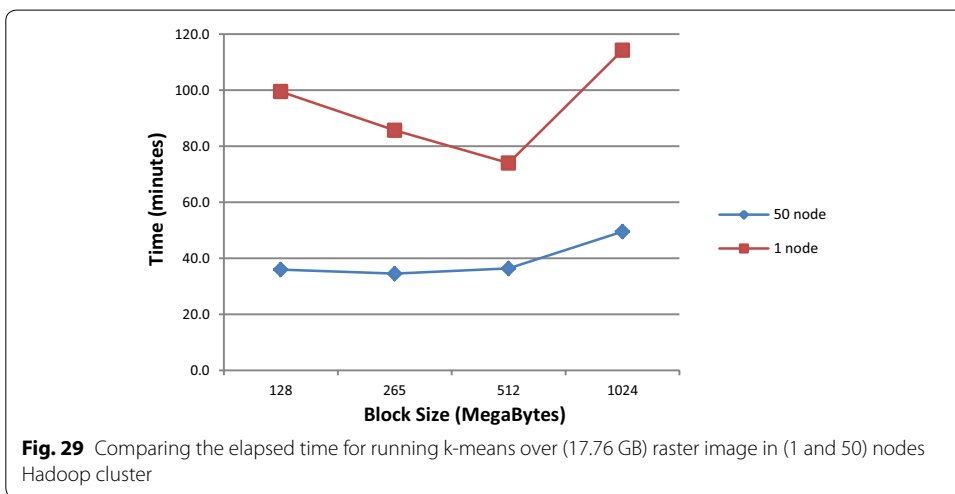
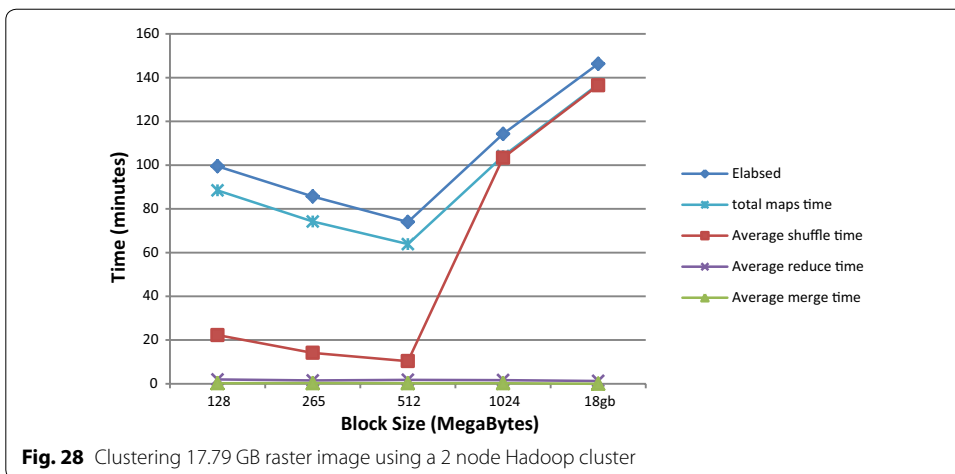


only four mappers can ever run for processing the data from this image. For a large Cluster, this will leave the resources unutilized and lead to an increase the execution time.

**K-means using 50 nodes Hadoop cluster for Image No. 2**

The second image (Image No. 2) is for the same geographical location but with different resolution. It is a 6-band raster image and each band has 24000 rows and 18000 columns. In an uncompressed form this image requires 4.83 GB storage with pixel depth of 2 bytes (24,000 rows × 18,000 columns × 6 bands × 2 bytes of data). After running the indexing method upon this binary file we get a plain text ASCII representation (an indexed image) of size 17.76 GB. For further processing, the initial set of centroids are provided (these are the same centroids which were provided for the previous image considering the same geographical location and to make sure that the k-means clustering technique receives the same input). The values presented in Table 5 and represented in Fig. 27 have been averaged by running the clustering technique four times upon a full cluster of 50 nodes. For each of the run, the block-size is changed and the file is updated to follow the new block size. As the current image is much larger than Image No. 1, the





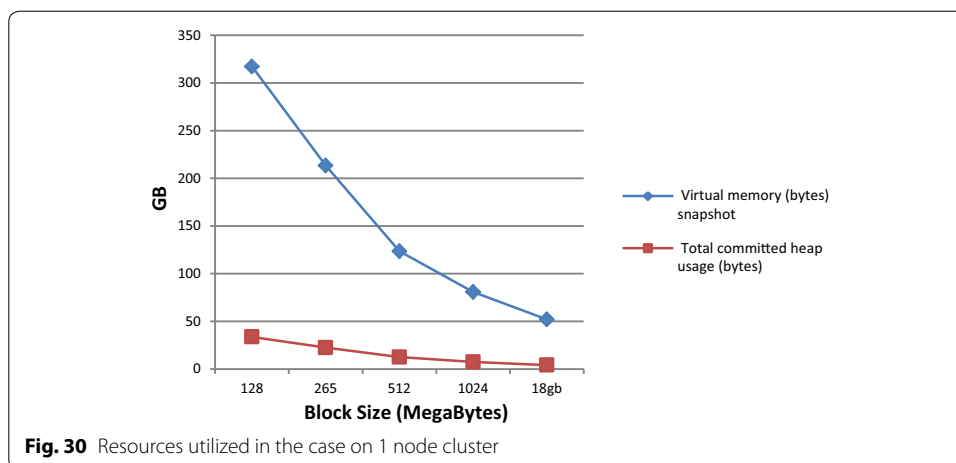
block size have also been increase accordingly. The block size for the experimentations used are 128 MB, 256 MB, 512 MB, and 1024 MB respectively.

**K-means using 2 node Hadoop cluster for Image No. 2**

From the values in Table 6 it can be observed that the most appropriate block size for the Image No. 2 is 512 MB which requires the minimum execution time. The values have been averaged over four consecutive runs to minimize error margin. For very large block size (of 18 GB), as represented in Fig. 28, the Shuffle time increases exponentially as the phase requires data to be transferred from one node to another and is heavily dependent on the network.

**Comparing results of K-means between 2 nodes and 50 nodes Hadoop cluster for Image No. 2**

From the results represented in Fig. 29, the advantage of utilizing the Hadoop distributed environment for processing large raster images is evident. It can be seen that the execution time decreases considerably when the same Image (No. 2) is processes upon a full cluster of 50 nodes rather than two nodes cluster. In Fig. 29, the red line represent



the time needed to cluster the image using two node or a Hadoop cluster of to machines whereas the blue line represent the time needed to do the same processes using 50 nodes. Even for larger block size, the results remain constant and as is evident.

Figure 29 also shows that running the algorithm using two nodes, for the dataset, the block size was 512 MB provided the best performance. While processing the same dataset using 50 machines, 256 MB block size is preferable. With a quick comparison of clustering Image No. 2 (size: 18 GB) and Image No. 1 (size: 1.9 GB), it is found that 32 MB and 64 MB block size provided the minimum execution time for 50 Node cluster and 2 node cluster respectively. It is concluded that the block size is an important factor to be considered for preparing the dataset along with other factors such as the number of nodes (machines) in the Hadoop cluster when deploying a geo-spatial Big Data processing framework.

The 50 nodes Hadoop cluster is configured with about 360 GB of memory. From Fig. 30, it is visible that a large amount of memory is being used with 128 MB of data blocks and this can be attributed to the fact that all of the data nodes in the cluster are contributing. The use of memory decreases with the increase in the data blocks because several of the data nodes remain idle and does not contribute as no blocks are available local to them for processing. Similar results have been obtained with Image No. 1 with block size ranging from 16 to 256 MB and have been excluded for brevity.

**Evaluating the results of clustering using the proposed technique with widely used geospatial image analysis software**

Table 7 represents the percentage of the number of pixels in each cluster from the two different images. As the images are for the same geographic area but with a different resolution, the results of clustering with the proposed techniques and using ENVI, a widely used image processing application, are identical. The percentage difference for both the images have been calculated and verified for multiple runs as described in the above sections. The percentage difference is found to be negligible except in case of Cluster No. 4 and Cluster No. 10 in the discussed case which validates the use of the proposed technique for multi-dimensional data.

**Table 7 Comparing the number of pixels for the important clusters**

Cluster no.	No. of pixels in Image No. 1	Cluster area (X%)	No. of pixels in Image No. 2	Cluster area (Y%)	No. of pixels in (ENVI results)	Cluster area (Z%)	Difference $(Z - (Y + X)/2)$
1	505,827	1.05	4,553,869	1.05	82,766	1.40	0.34%
2	1,807,105	3.76	16,300,000	3.77	229,199	3.87	0.10%
3	1,989,750	4.15	17,900,000	4.14	304,015	5.14	0.99%
4	3,055,365	6.37	27,500,000	6.37	534,589	9.03	2.66%
5	4,917,089	10.24	44,300,000	10.25	582,282	9.84	0.41%
6	5,598,346	11.66	50,300,000	11.64	647,603	10.94	0.71%
8	6,484,568	13.51	58,400,000	13.52	822,424	13.89	0.38%
7	6,819,688	14.21	61,400,000	14.21	854,459	14.43	0.22%
9	7,689,434	16.02	69,200,000	16.02	896,664	15.15	0.87%
10	9,132,838	19.03	82,200,000	19.03	965,999	16.32	2.71%
Total	48,000,000 (~48 Megapixels)	100	432,000,000 (~0.43 Gigapixels)	100	5,920,000 (~6 Megapixels)	100	0.00

### Conclusion and future work

The main objective of the present work is to utilize the advances in distributed processing, specifically MapReduce programming paradigm, to facilitate big geospatial data processing and mining. The multispectral essence of the raster images has been preserved by converting multi-spectral space to multi-dimensional geometric space. The existing data mining and machine learning techniques are limited by scale and the ones even which are available for use in a distributed environment for processing raster data and scalability only support processing of single spectral band at a time. With the development of this work, it has now become easier to port existing image processing techniques for distributed processing of giga-pixel imagery and application of custom logic for development of applications.

A big geospatial data mining platform based on Apache Hadoop distributed processing environment has been developed in this work. The developed mining platform comes with a group of editing, filtering and other image processing techniques to help better extracting the geographical features out of the image data. The processing (mining) capabilities and other image processing facilities of the framework have been tested using several images of unconventional size in scale of giga-pixels, and it shows the advantages of using the developed framework upon a distributed environment as compared to a desktop GIS application using a single machine. In a distributed environment, clustering was performed on sample images over 50 node (machines) cluster and over 2 nodes (machines). ENVI, a desktop GIS application, was also used over the same image of a reduced resolution to analyze the results of the proposed clustering technique. There is a gain of factor of about 2 to 2.5 in time of data process depending on the block size employed. The clustering results comparison shows maximum deviation of 2.7 percent which is negligible. The analysis of times of various sub processes shows clearly advantages of this processing in MapReduce programming. The classified image data can be used further for spatial as well as temporal characterizing the geospatial objects in an area for scene modelling and also modelling geo-spatial processes.

The results of clustering have been tested by going from spatial to geometrical space and similarly other methods can also be adapted to support processing of multiband data coherently.

The results derived also highlight the importance of requirement of compute, memory, storage and network infrastructure for processing such large datasets. Appropriate data storage mechanisms are also required for fast access to large amounts of data as in a distributed environment the data is distributed across a number of nodes. The nodes where the data blocks are stored contribute to the overall performance of the system. This has also been evaluated by using different block size when storing the data.

The proposed work can be extended to support other spatial mining techniques. Distributed processing techniques developed for clustering can be extended to support other types of processing. Workflows are an important component of any geospatial data mining systems and the current system can be extended to support workflow type of applications. The current system does not have a visualization interface which can be provided and will allow to see big geospatial data at various levels of abstractions. A workflow pipeline which can insert the data into a database for an application server such as GeoServer is highly desirable and is currently being worked out.

#### Abbreviations

GIS: geographic information system; GeoTIFF: georeferenced tagged image file format; GB: gigabyte; LON: longitude; LAT: latitude; BSQ: band sequential; BIP: band interleaved by pixel; BIL: band interleaved by line; SPOT: Satellite Pour l'Observation de la Terre; GS-Hadoop: Geo Spatial Hadoop; IPS: information processing systems; MDB: multidimensional databases; OLAP: on-line analytical processing; OLTP: online transactional processing; KDD: knowledge discovery in databases; DMLC: data mining life cycle; HDFS: Hadoop Distributed File System; COTS: commercially of the shelf; KNN: K-Nearest Neighbours.

#### Acknowledgements

We are grateful to Shri T. P. Singh, Director, BISAG for his keen interest in and support to this work. We are also grateful to Apache Software Foundation and the Open Source community for making a plethora of softwares open source and without the availability of which the current development would not have been possible.

#### Authors' contributions

MA has developed the proposed mining platform development of the required programs and test cases, analysis and interpretation of data and drafting of the manuscript. AJ has served as advisor in study conception and for critical revision. MBP has also served as advisor, and critically reviewed the complete developments. All authors read and approved the final manuscript.

#### Funding

None.

#### Availability of data and materials

The data set used in this paper is derived from Landsat 8 multispectral images for the location of Aravalli Fort Hills, an area North of Gujarat state, India with Geographic Location (around): Lat: 24° 00' North, and Lon: 72° 54' East). The data-set has been processed and provided by BISAG, Gandhinagar, Gujarat, India.

#### Competing interests

The authors declare that they have no competing interests.

#### Author details

<sup>1</sup> Administrative Sciences College Hadhramout, University of Science and Technology, Hadhramout, Yemen. <sup>2</sup> Bhaskaracharya Institute for Space Applications and Geo-Informatics, Gandhinagar 382007, India.

Received: 30 April 2019 Accepted: 22 August 2019

Published online: 05 September 2019

#### References

1. Ahmad A, Dey L. A k-mean clustering algorithm for mixed numeric and categorical data. *Data Knowl Eng.* 2007;63:503–27.
2. Alarabi L, Mokbel MF, Musleh M. St-hadoop: a mapreduce framework for spatio-temporal data. *Geoinformatica.* 2018;22:785–813.



3. Alkathiri M, Jhummarwala A, Potdar M. Geo-spatial big data mining techniques. *Int J Comput Appl.* 2016;135:28–36.
4. Bédard Y, Merrett T, Han J. Fundamentals of spatial data warehousing for geographic knowledge discovery. *Geogr Data Min Knowl Discov.* 2001;2:53–73.
5. Bennett J. *OpenStreetMap*. Birmingham: Packt Publishing Ltd.; 2010.
6. Bereta K, Koubarakis M. Ontop of geospatial databases. In: *International semantic web conference*. Cham: Springer; 2016. p. 37–52.
7. Bernard E, Naveau P, Vrac M, Mestre O. Clustering of maxima: spatial dependencies among heavy rainfall in France. *J Clim.* 2013;26:7929–37.
8. Bhosale HS, Gadekar DP. A review paper on big data and hadoop. *Int J Sci Res Publ.* 2014;4:1.
9. Borodin A, Mirvoda S, Porshnev S. Analysis of multidimensional data with high dimensionality: data access problems and possible solutions. In: *ITM web of conferences*. Les Ulis: EDP Sciences; 2016. p. 01005.
10. Borthakur D. The hadoop distributed file system: architecture and design. *Hadoop Proj Website.* 2007;11:21.
11. Borthakur D. HDFS architecture guide. Hadoop apache project. 2008. <http://hadoop.apache.org/common/docs/current/hdfsdesign.pdf>. p. 39.
12. Bradley PS, Fayyad UM, Reina C. Scaling clustering algorithms to large databases. *KDD.* 1998;98:9–15.
13. Calimeri F, Caracciolo M, Marzullo A, Stamile C. BioHIP: biomedical hadoop image processing interface. In: *International workshop on machine learning, optimization, and big data*. Cham: Springer; 2017. p. 540–8.
14. Campbell JB, Wynne RH. *Introduction to remote sensing*. New York: Guilford Press; 2011.
15. Council NR. *Landsat and beyond: sustaining and enhancing the nation's land imaging program*. Washington, D.C: National Academies Press; 2013.
16. Coveney M. Corporate performance management (CPM). 2003. <http://www.businessforum.com/Comshare04/B.html>.
17. Crema S, Cavalli M. SedInConnect: a stand-alone, free and open source tool for the assessment of sediment connectivity. *Comput Geosci.* 2018;111:39–45.
18. Dagade V, Lagali M, Avadhani S, Kalekar P. Big data weather analytics using hadoop. *Int J Emerg Technol Comput Sci Electron (IJETCSE).* 2015;14:0976–1353.
19. de Smith MJ, Goodchild MF, Longley P. *Geospatial analysis: a comprehensive guide to principles, techniques and software tools*. Leicester: Troubador Publishing Ltd; 2009.
20. Ding Q, Khan M, Roy A, Perrizo W. The P-tree algebra. In: *Proceedings of the 2002 ACM symposium on applied computing*. ACM; 2002. p. 426–31.
21. EARTH\_OBSERVATION\_SYSTEM. 2019. EOS processing—classic gis algorithms. <https://eos.com/eos-processing/>.
22. Egenhofer MJ, Herring J. Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical report, Department of Surveying Engineering, University of Maine, Orono, ME; 1990.
23. Eldawy A, Mokbel MF. Spatialhadoop: a mapreduce framework for spatial data. In: *2015 IEEE 31st international conference on data engineering (ICDE)*. New York: IEEE; 2015. p. 1352–63.
24. Eldawy A, Niu L, Haynes D, Su Z. Large scale analytics of vector + raster big spatial data. In: *Proceedings of the 25th ACM SIGSPATIAL international conference on advances in geographic information systems*. New York: ACM; 2017. p. 62.
25. Evans MR, Oliver D, Yang K, Zhou X, Ali RY, Shekhar S. Enabling spatial big data via CyberGIS: challenges and opportunities. *CyberGIS for geospatial discovery and innovation*. Dordrecht: Springer; 2019.
26. Foundation AS. *Apache hadoop*. The Apache Software Foundation. 2018. <https://hadoop.apache.org/>.
27. Ghazi MR, Gangodkar D. Hadoop, MapReduce and HDFS: a developers perspective. *Procedia Comput Sci.* 2015;48:45–50.
28. Gopalani S, Arora R. Comparing apache spark and map reduce with performance analysis using k-means. *Int J Comput Appl.* 2015;113:8–11.
29. Goward SN, Masek JG, Williams DL, Irons JR, Thompson R. The Landsat 7 mission: terrestrial research and applications for the 21st century. *Remote Sens Environ.* 2001;78:3–12.
30. Haklay M, Weber P. Openstreetmap: user-generated street maps. *IEEE Pervas Comput.* 2008;7:12–8.
31. Jhummarwala A, Mazin A, Potdar M. Geospatial hadoop (GS-Hadoop) an efficient mapreduce based engine for distributed processing of shapefiles. In: *Proceedings of the the 2nd international conference on advances in computing, communication, & automation*, Bareilly, India; 2016. p. 1–7.
32. Jo J, Lee K-W. High-performance geospatial big data processing system based on MapReduce. *ISPRS Int J Geo-Inf.* 2018;7:399.
33. Johnson L. Fiber optics: mature and growing fast. *Tech Dir.* 2016;76:22.
34. Keim DA, Panse C, Sips M, North SC. Pixel based visual data mining of geo-spatial data. *Comput Gr.* 2004;28:327–44.
35. Koperski K. *A progressive refinement approach to spatial data mining*. Canada: Simon Fraser University; 1999.
36. Lauer DT, Morain SA, Salomonson VV. The Landsat program: its origins, evolution, and impacts. *Photogramm Eng Remote Sens.* 1997;63:831–8.
37. Lausch A, Schmidt A, Tischendorf L. Data mining and linked open data—new perspectives for data analysis in environmental research. *Ecol Model.* 2015;295:5–17.
38. Lenka RK, Barik RK, Gupta N, Ali SM, Rath A, Dubey H. Comparative analysis of SpatialHadoop and GeoSpark for geospatial big data analytics. In: *2016 2nd international conference on contemporary computing and informatics (IC3I)*. New York: IEEE; 2016. p. 484–8.
39. Mennis J, Guo D. Spatial data mining and geographic knowledge discovery—an introduction. *Comput Environ Urban Syst.* 2009;33:403–8.
40. Mukherjee A, Datta J, Jorapur R, Singhvi R, Haloi S, Akram W. Shared disk big data analytics with apache hadoop. In: *2012 19th international conference on high performance computing*. New York: IEEE; 2012. p. 1–6.
41. Murray AT, Shyy T-K. Integrating attribute and space characteristics in choropleth display and spatial data mining. *Int J Geogr Inf Sci.* 2000;14:649–67.
42. Ng RT, Han J. CLARANS: a method for clustering objects for spatial data mining. *IEEE Trans Knowl Data Eng.* 2002;14:1003–16.

43. Nijmeijer R, de Haas A, Dost R, Budde P. ILWIS 3.0 academic: user's guide; 2001.
44. Ooi B, Sacks-Davis R, Han J. Indexing in spatial databases. Unpublished/Technical Papers; 1993.
45. Parker JR. Extracting vectors from raster images. *Comput Gr*. 1988;12:75–9.
46. Peralta D, del Río S, Ramírez-Gallego S, Triguero I, Benitez JM, Herrera F. Evolutionary feature selection for big data classification: a mapreduce approach. *Math Probl Eng*. 2015. <https://doi.org/10.1155/2015/246139>.
47. Rabbitt MC. The United States geological survey, 1879–1989, US Government Printing Office. 1989.
48. Samson G, Lu J, Xu Q. Large spatial datasets: present challenges, future opportunities. In: Proceedings of the international conference on change, innovation, informatics and disruptive technology ICCIIDT'16, London-UK, October 11, 12, 2016; 2016. p. 204–17.
49. Samson GL, Lu J, Wang L, Wilson D. An approach for mining complex spatial dataset. In: Proceedings of the international conference on information and knowledge engineering (IKE), 2013. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp). p. 1.
50. Sarode AJ, Mishra A. Audit and analysis of impostors: an experimental approach to detect fake profile in online social network. In: Proceedings of the sixth international conference on computer and communication technology 2015. New York: ACM; 2015. p. 1–8.
51. Shahid R, Bertazzon S, Knudtson ML, Ghali WA. Comparison of distance measures in spatial analytical modeling for health service planning. *BMC Health Serv Res*. 2009;9:200.
52. Sloan KR, Tanimoto SL. Progressive refinement of raster images. *IEEE Trans Comput*. 1979;28:871–4.
53. Szeliski R. Computer vision: algorithms and applications. New York: Springer Science & Business Media; 2010.
54. Talbot D, Warner E, Anderson C, Hessekiel K, Jones D. A Massachusetts municipal light plant seizes internet access business opportunities; 2015.
55. Trujillo J, Palomar M. An object oriented approach to multidimensional database conceptual modeling (OOMD). In: Proceedings of the 1st ACM international workshop on data warehousing and OLAP. New York: ACM; 1998. p. 16–21.
56. Uzunkaya C, Ensari T, Kavurucu Y. Hadoop ecosystem and its analysis on tweets. *Procedia-Soc Behav Sci*. 2015;195:1890–7.
57. van de Weijer J. Local mode filtering J. van de Weijer R. van den Boomgaard Intelligent Sensory Information Systems Faculty of Science, University of Amsterdam Kruislaan 403, 1098 SJ Amsterdam, The Netherlands.
58. Vatsavai RR, Ganguly A, Chandola V, Stefanidis A, Klasky S, Shekhar S. Spatiotemporal data mining in the era of big spatial data: algorithms and applications. In: Proceedings of the 1st ACM SIGSPATIAL international workshop on analytics for big geospatial data; 2012. p. 1–10.
59. Wagstaff K, Cardie C, Rogers S, Schrödl S. Constrained k-means clustering with background knowledge. In: *ICML*; 2001. p. 577–84.
60. Wang C-S, Lin S-L, Chang JY. MapReduce-based frequent pattern mining framework with multiple item support. In: Asian conference on intelligent information and database systems. New York: Springer; 2017. p. 65–74.
61. Wang W, Yang J, Muntz R. STING: a statistical information grid approach to spatial data mining. In: *VLDB*; 1997. p. 186–195.
62. Wang W, Yang J, Muntz R. STING+: an approach to active spatial data mining. In: Proceedings 15th international conference on data engineering (Cat. No. 99CB36337). New York: IEEE; 1999. p. 116–125.
63. Witten IH, Frank E, Hall MA, Pal CJ. Data mining: practical machine learning tools and techniques. Burlington: Morgan Kaufmann; 2016.
64. Xiaoke Z, Chao M, Haifeng H, Fangfang L. Radiometric correction based on multi-temporal spot satellite images. In: 2009 international conference on wireless communications & signal processing. New York: IEEE; 2009. p. 1–6.
65. Yao X. Research issues in spatio-temporal data mining. In: Workshop on geospatial visualization and knowledge discovery, University Consortium for Geographic Information Science, Virginia; 2003. p. 1–6.
66. Yoon I, Yi S, Oh C, Jung H, Yi Y. Distributed video decoding on hadoop. *IEICE Trans Inf Syst*. 2018;101:2933–41.
67. Zhang Z, Zhang J, Xue H. Improved K-means clustering algorithm. In: 2008 congress on image and signal processing. New York: IEEE; 2008. p. 169–72.
68. Zhao T, Zhang C, Anselin L, Li W, Chen K. A parallel approach for improving Geo-SPARQL query performance. *Int J Digit Earth*. 2015;8:383–402.
69. Zhao W, Ma H, He Q. Parallel k-means clustering based on mapreduce. In: IEEE international conference on cloud computing. New York: Springer; 2009. p. 674–9.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.