Journal of Geographical Systems (2024) 26:73–90
https://doi.org/10.1007/s10109-023-00423-y

ORIGINAL ARTICLE

Accelerated multi-hillshade hierarchic clustering for automatic lineament extraction

Ondřej Kaas[1,3] · Jakub Šilhavý[2,3] · Ivana Kolingerová[1,3] · Václav Čada[2,3]

Received: 13 August 2020 / Accepted: 2 August 2023 / Published online: 30 September 2023

Abstract
The lineaments are linear features reflecting mountain ridges or discontinuities in the geological structure. Lineament extraction is not an easy problem. Recently, an automatic approach based on multi-hillshade hierarchic clustering (MHHC) has been developed; the approach is based on line extraction from a raster image. An essential part of this approach is spatial line segment clustering, a powerful but relatively slow tool. This paper presents a modification of MHHC, which solves the spatial line segment clustering as a facility location problem. The proposed modification is faster than MHHC while not changing the method's core.

Keywords Lineaments · GIS · Automatic extraction · Clustering · Facility location

1 Introduction

A lineament is a mappable, simple, or composite linear feature of a surface whose parts are aligned in a rectilinear or slightly curvilinear relationship and which differs distinctly from the patterns of adjacent features and presumably reflects a subsurface phenomenon. Discontinuity in a geological structure, a mountain ridge, rock boundaries, sedimentary layers, wetness, vegetation changes, and many more can form a lineament (see O'Leary et al. 1976). In the field of lineament extraction from digital images, the lineament is seen as a line or a linear formation that is in strong contrast to background pixels (Soto-Pinto et al. 2013; Hashim et al. 2013). The lineaments are beneficial for both application and research areas. The

✉ Ivana Kolingerová
kolinger@kiv.zcu.cz

[1] Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 306 14 Pilsen, Czech Republic

[2] Department of Geomatics, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 306 14 Pilsen, Czech Republic

[3] NTIS - New Technologies for the Information Society, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 306 14 Pilsen, Czech Republic

landslide hazard assessment, monitoring of seismic and volcanic activities, and natural resource exploration are examples of application use (Ramli et al. 2010; Mallast et al. 2011). Geological faults comprise an exciting and important subject of study in geosciences; therefore, identifying the linear patterns of geological fault structures becomes an important task (Panagiotakis and Kokinou 2014). In geomorphological research, lineaments can be considered as an expression of morphotectonic fields, which allows for studying temporal changes of tectonic actions on land surface (Minár and Sládek 2009) or understanding of the genetic and spatial relationships of fracture systems (Seleem 2013).

Lineament extraction is a method producing line segments as results. Due to the fuzzy character of lineaments, the input data are usually multiple line segments with some inaccuracy which generally means that several similar line segments represent one lineament. The human eye easily recognizes these objects, but computer identification is not trivial (see Fig. 1).

The extraction of lineaments is often done manually. The disadvantage of the manual approach is its subjectivity—the user's opinion influences the result. What is more, manual extraction is time-consuming. Therefore, semi-automatic and automatic approaches have been developed. A step often used in non-manual solutions is the spatial clustering of line segments. It is a method that groups line segments based on their length, slope, and position similarity. A representative line segment of each cluster is picked as a lineament. The spatial clustering enables more data input files to process, thus providing results of higher quality; however, time consumption might be too high.

An automatic representative method of this type is the multi-hillshade hierarchic clustering (MHHC) (Silhavy et al. 2016), which is artifact-resistant, robust, and provides similar results as manual extraction; however, it is relatively slow. Incorporating a more sophisticated computer science approach may bring a substantial acceleration while keeping good properties of the original method. And this is the main focus of this paper. We concentrate on the clustering step, incorporating a facility-location-based approach. The resulting algorithm, accelerated multi-hillshade hierarchic clustering for automatic lineament extraction (AMHHC), is faster than the original while keeping its required lineament-extraction behavior.

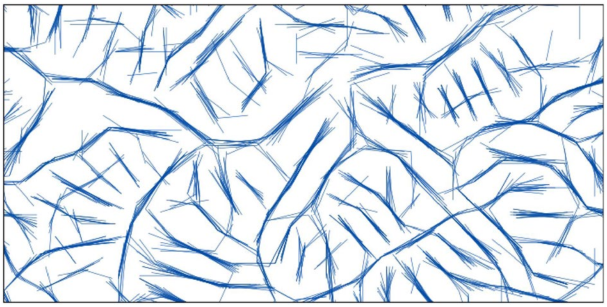


Fig. 1 Example of input line segments for lineament extraction

The paper is structured as follows. Section 2 covers state of the art. Sections 3 and 4 are devoted to the facility location problem and the original multi-hillshade hierarchic clustering algorithm, which are fundamentals for the proposed solution. The description of the newly developed approach follows in Sect. 5. Results, experiments, and comparison are in Sect. 6. Section 7 concludes the paper.

2 State of the art

Clustering is a widely used principle in computer science (see Dubes and Jain 1980 for a survey). It is a process of grouping similar elements into so-called clusters, while elements from different clusters must be as dissimilar as possible (Likas et al. 2003). The input data to be clustered are usually spatial points, but images, patterns, or geometrical objects are also possible. Their suitable representation for clustering is the points in D-dimensional space, represented by its D spatial coordinates. The definition of similarity is crucial and is measured with a distance function, a metric. In the simplest case, the Euclidean distance is used. Many metrics (or quasi-metrics) for special clustering tasks have been developed, such as Mahalanobis metrics for detecting hyper-ellipsoidal clusters (Mao and Jain 1996), Itakura-Saito distance for vector quantization in speech processing (Linde et al. 1980), L1 distance for pattern recognition (Kashima et al. 2008) or Bregman distances used in machine learning (Banerjee et al. 2005).

The clustering problem is classified as NP-hard, so the algorithms produce only approximate results. The quality of the approximation is counterbalanced with the time of computation. Clustering techniques are applied in many areas, such as data analyses (Dubes and Jain 1980), pattern recognition (Baraldi and Blonda 1999), image processing (Jain and Flynn 1996), information retrieval (Rasmussen 1992), marketing (Russell and Lodwick 1999) or biology (Legendre and Legendre 2012).

Clustering methods can be subdivided into several ways. The taxonomy survey of clustering can be found in (Jain and Dubes 1988).

K-means (MacQueen et al. 1967) and DBSCAN (density-based spatial clustering of applications with noise, Ester et al. 1996) are well-known partitioning (i.e., non-hierarchical) algorithms. K-means is used for its simplicity and performance and is supported by many computing libraries. The most significant disadvantage of K-means is that the number of clusters must be defined before the algorithm starts.

DBSCAN works well for connected regions of sufficiently high density; otherwise, the data are marked as noise. For the DBSCAN use, two parameters ε and MinSample must be defined; ε specifies the maximum allowable distance of points from the cluster center. The cluster is formed at the point if at least MinSample points are at a maximal distance ε from this point. In order to improve the DBSCAN time complexity $\mathcal{O}(N^2)$, it is necessary to involve spatial index structures, such as R^* tree (Beckmann et al. 1990), kd-tree (Bentley 1975) or at least a grid, then time complexity $\mathcal{O}(NlogN)$ can be achieved. The algorithm does not work well for the variable density of input points.

Revisions of existing studies proposing lineament extraction using line segment clustering are as follows. Mallast et al. (2011) used a singularization of identical results from two automatic extraction algorithms to improve the accuracy of the lineament map. The whole process was made using GIS analysis but without any automation. The method in Jordan and Schott (2005) processed eight different input files to produce eight lineament maps that were overlaid, and the line replicates were manually eliminated.

Other authors (Abdullah et al. 2010; Seleem 2013) work with multiple input files to extract lineaments; they merge the input data to obtain one result.

More references to other methods in the lineament extraction area, such as (Vaz 2011), (Masoud and Koike 2011), and (Soto-Pinto et al. 2013), can be found in (Silhavy et al. 2016). Let us briefly present at least some, namely those which have appeared since the publication of the mentioned paper.

Bonetto et al. (2015) detect linear features on a DTM utilizing algorithms based on principal curvature values. Then, the features are grouped according to data collected from the literature review regarding the expected orientation of lineaments in the studied area.

The paper by Thiele et al. (2017) describes a novel least-cost path method that can "follow" structure traces and lithological contacts between user-defined control points in both 2D gridded datasets (photographs, geophysical imagery, etc.) and dense 3D point clouds (virtual outcrop models). The method is suitable for computer-assisted digitizing structural traces in the point cloud, image, and raster datasets.

Vasuki et al. (2017) is an interactive image segmentation algorithm that harnesses the geologist's input and exploits automated image analysis to provide a practical tool for lithology boundary detection, using photographic images of rock surfaces. The user is expected to draw rough markings to indicate the locations of different geological units in the image. Image segmentation is performed by segmenting regions based on their homogeneity in color.

Adiri et al. (2017) compares ASTER, Landsat-8, and Sentinel 1 data sensors in automatic lineament extraction, combining image data with pre-existing geological maps, the digital elevation model (DEM), and the ground truth.

Karimi and Karimi (2017) extracts line segments from the image, and cluster analysis is based on (Silhavy et al. 2016) with some smaller differences.

Msaddek et al. (2019), a semi-automatic extraction of lineaments using satellite imagery of Landsat-8 OLI (Operational Land Imager) and digital elevation model of SRTM (Shuttle Radar Topography Mission) is presented. The developed algorithm used for this purpose combines STA (segment tracing algorithm), ALEGHT (automatic lineament extraction by generalized Hough transform) and AERA (alluvial effect reducing algorithm).

Yeomans et al. (2019) presented two complementary but stand-alone OBIA methods for lineament detection (top-down and bottom-up), which both enable semi-automatic regional lineament mapping.

In the work by Mohammadpour et al. (2020), the CANNY algorithm was first employed as an edge-detector filter. Later, Hough transform was used to extract linear features from satellite imagery and geophysical magnetometry data.

The paper by Xu et al. (2020) uses wavelet modulus maximum transformation to detect the edges with four scales on Landsat-8 OLI B5 image and analyzes their multi-scale characteristics. The incomplete lineaments are extracted using the 2D Otsu algorithm. Secondly, the hillshade map generated based on DEM is processed to generate a binarized linear shadow superimposed on the lineaments preliminarily extracted to obtain the optimized lineaments.

The study presented in Salui (2018) provides a decision on using a reliable method and data source for automated lineament extraction.

The following papers are more oriented to software than to the used method: The paper by Masoud and Koike (2017) describes the LINDA tool (LINeament Detection and Analysis), a software developed in Visual Basic, which automates processes of detection and analysis of linear features from grid data of topography (digital elevation model, DEM), gravity and magnetic surfaces, as well as data from remote sensing imagery. NetworkGT (Nyberg et al. 2018) is an open-source toolbox for ArcGIS capable of efficient sampling, analysis, and spatial mapping of geometric and topological attributes of two-dimensional fracture networks. The toolbox helps to extract and plot geometric and topological information from a given two-dimensional fracture network, including rose diagrams, frequency distribution plots and topology, and maps of topological parameters. The paper by Rahnama and Gloaguen (2014) and its later continuation provide a MATLAB-based toolbox TecLines (Tectonic Lineament Analysis) for locating and quantifying lineament patterns using satellite data and digital elevation models.

The algorithm TRACLUS based on a partition-and-group framework presented in Lee et al. (2007) was developed for clustering trajectories, the problem slightly similar to the lineament extraction. The algorithm contains two phases, trajectory partitioning, and line segment clustering. The clustering part is based on the DBSCAN algorithm; therefore, it does not produce good clustering results for data sets with significant differences in densities since the MinSample-ε combination cannot then be chosen appropriately for all clusters or easily determined before clustering (the algorithm uses a heuristic to set the parameters properly). For satisfactory results, the algorithm requires connected regions of sufficiently high density; otherwise, the data are marked as noise. Therefore, while working well for trajectories, the algorithm is unsuitable for the lineaments problem.

The first algorithm for automatic clustering in the lineament extraction problem, called multi-hillshade hierarchic clustering (MHHC), was done by Silhavy et al. (2016). The algorithm clusters lineaments to singularize results from multiple different input files. The algorithm provides good results; however, it is relatively slow. Therefore, this paper proposes a modification called accelerated multi-hillshade hierarchic clustering (AMHHC), which preserves the MHHC's good properties and is faster. The acceleration is achieved by more efficient line segment clustering, understood as an instance of the facility location problem, solved by local search.

The following two sections cover the Local Search algorithm and MHHC as two fundamental sources of the proposed solution.

3 Facility location and local search

The clustering used in the proposed algorithm is based on the facility location problem solution. Let us describe the essence of facility location before proceeding to a more formal definition.

Let us have a set of points (sometimes called clients) and a set of facilities (usually also points, sometimes chosen from the set of clients). The task is to attach each particular point to one facility so that some evaluation function is optimized. A typical evaluation function is the sum of the distances of the points from their facilities to be minimized.

The problem is too difficult to find the global optimum, so local techniques and various heuristics are used. For example, first, some initial solution is generated and then is tried to be optimized by local reassignments of the points between facilities, the so-called local search (Korupolu et al. 1998; Charikar and Guha 1999).

The more formal description of the facility location as solved by the local search is as follows (Skála and Kolingerová 2011).

Let $F = \{\boldsymbol{f}_i, i = 1, 2, \dots, K\}$ be a set of facilities and $C = \{\boldsymbol{c}_j, j = 1, 2, \dots, N\}$ be a set of clients—the points to be allocated to facilities. A new facility $\boldsymbol{f}_{\boldsymbol{c}_j}$ can be opened at the client $\boldsymbol{c}_j$, but a facility cost t_{cost} has to be paid for opening a new facility. The goal is to minimize the overall clustering cost Q, defined as

$$Q = K \cdot t_{\mathrm{cost}} + \sum_{i=1}^{K} \sum_{j \in \boldsymbol{f}_i} d(\boldsymbol{c}_j, \boldsymbol{f}_i) \tag{1}$$

where $d(\boldsymbol{c}_j, \boldsymbol{f}_i)$ is the distance between a point $\boldsymbol{c}_j \in C$ and its facility $\boldsymbol{f}_i \in F$, measured by a metrics.

A high value of the facility cost brings a low number of large clusters and vice versa.

First, a coarse initial solution is found (Meyerson 2001). The first client is taken as a facility; the other clients are then taken in random order. A point $\boldsymbol{c}_j$ is connected to the closest already open facility $\boldsymbol{f}_i$ based on the distance $d(\boldsymbol{c}_j, \boldsymbol{f}_i)$. A new facility $\boldsymbol{f}_{\boldsymbol{c}_j}$ is opened at the point $\boldsymbol{c}_j$ if $d(\boldsymbol{c}_j, \boldsymbol{f}_i) > t_{\mathrm{cost}}$. Otherwise, it is opened with probability $d(\boldsymbol{c}_j, \boldsymbol{f}_i)/t_{\mathrm{cost}}$. This initial coarse solution is improved by further iterations of the following local search step.

The local search step randomly selects a point $\boldsymbol{c}_p \in C$. For $\boldsymbol{c}_p$ it is computed whether a new facility $\boldsymbol{f}_{\boldsymbol{c}_p}$ if opened here could improve the current solution (if $\boldsymbol{f}_{\boldsymbol{c}_p}$ does not exist, a facility cost t_{cost} would have to be paid for its opening). The points closer to $\boldsymbol{f}_{\boldsymbol{c}_p}$ than to their current facility can be reassigned. Due to this reassignment, some facilities may become too small and be closed to spare their facility costs.

A possible improvement to the current solution by declaring the point $\boldsymbol{c}_p$ a new facility $\boldsymbol{f}_{\boldsymbol{c}_p}$ and reassigning all near points from their facilities to $\boldsymbol{f}_{\boldsymbol{c}_p}$ is measured by a gain function according to the following relation:

$$\text{gain}(p) = -t_{\text{cost}} + \sum_{j=1}^{N} ds_j + \sum_{i=1}^{K} sp_i \tag{2}$$

where t_{cost} is the facility cost (zero if the facility $\boldsymbol{f}_{\boldsymbol{c}_p}$ is already open), ds_j is the distance spared by reassigning the client $\boldsymbol{c}_j$ from its current facility to the facility candidate $\boldsymbol{f}_{\boldsymbol{c}_p}$, sp_i (close spare) is the facility cost minus expenses for reassigning all remaining points from their current facility $\boldsymbol{f}_i$ to $\boldsymbol{f}_{\boldsymbol{c}_p}$.

If the current facility $\boldsymbol{f}_i$ lies closer to c_j than $\boldsymbol{f}_{\boldsymbol{c}_p}$, then $ds_j < 0$ and ds_j are set to zero. Similarly, if $sp_i < 0$ (facility $\boldsymbol{f}_i$ has enough points and, therefore, no spare can be achieved by its closure and reassigning all its points to the new facility $\boldsymbol{f}_{\boldsymbol{c}_p}$), then sp_i is set to zero.

If $\text{gain}(p) > 0$, the facility $\boldsymbol{f}_{\boldsymbol{c}_p}$ at the point $\boldsymbol{c}_p$ is opened (if not already open), and reassignments and closures are performed.

The number of iterations of the local search step needed to come to a good clustering solution is at least $\mathcal{O}(NlogN)$ (Charikar and Guha 1999).

Advantages of the facility location/local search use for lineaments in comparison with DBSCAN are the possibility to cluster regardless of the density of the area, no need to build a space index to run fast, and the possibility to distinguish line segments with substantially different angles without joining them into one lineament.

4 Cluster line analysis in MHHC

In the following section, the line clustering in the original MHHC is described. More details can be found in Silhavy et al. (2016).

The goal is to group the input line segments into clusters and then filter clusters of small size and compute a representative line segment for each cluster. This representative line segment has the length, orientation, and spatial position calculated as the average from the line segments forming the same cluster. The average line segment is the required lineament, and the resulting set of average line segments is the required output set of extracted lineaments, the algorithm's goal.

Let us have a set L of line segments $\boldsymbol{l}_j, j = 1, 2, \ldots N$, where $\boldsymbol{l}_j$ is described by the following vector:

$$\boldsymbol{l}_j = [X_j, Y_j, \alpha_j, \lambda_j]^T \tag{3}$$

where X_j, Y_j are coordinates of the start vertex of the line segment, α_j is the angle between the line segment and the positive x-axis (*azimuth*) and λ_j is the length of the line segment.

The method needs four user-given parameters. The minimum number of line segments that are allowed to form a cluster is denoted as count threshold (ξ_{cnt}), the maximum angle between line segments in a cluster is called the *azimuth threshold* (ξ_{az}) and the minimal and maximal allowed size of the so-called buffer zone is $bfs_{\max}$

and $bfs_{\min}$, respectively. The parameters are explained below, with proper values in Sect. 6.

The process is shown in more detail in Algorithm 1. The first step (Alg.1, line 1) is the filtration of the input line segments to reduce the amount of processed data. The filtration is based on a grid approach. The line dataset is inserted into a grid, and each line segment increments a counter in the (even partially) covered cell. A higher counter value in the grid cell indicates more line segments in the same cell. The algorithm deletes all the line segments which cover the area with the average cell counter value lower than the given count threshold parameter ξ_{cnt}.

```
Algorithm 1: MHHC - cluster analysis
   Data: Array of line segments $L = \{\boldsymbol{l}_j, j = 1, 2, ..., N\}$, the minimal and
         maximal buffer zone size $bfs_{min}$ and $bfs_{max}$, the azimuth threshold
         $\xi_{az}$ and the threshold count $\xi_{cnt}$
   Result: ALS - list of representative (average) line segments
 1 Reduce noise using grid;
 2 Sort the line segments in $L$ according to their length decreasingly;
 3 $j \leftarrow 0$;
 4 while $L$ not empty do
 5     Take the current $\boldsymbol{l}_j$ from $L$. // The longest yet unprocessed line segment;
 6     Make a buffer zone about $\boldsymbol{l}_j$ using Eq. 4;
 7     Add all line segments lying completely within the buffer zone into a
         temporary cluster $\boldsymbol{f}_{\boldsymbol{l}_j}$;
 8     Filter the line segments in $\boldsymbol{f}_{\boldsymbol{l}_j}$ by the $\xi_{az}$;
 9     if the number of line segments in $\boldsymbol{f}_{\boldsymbol{l}_j} > \xi_{cnt}$ then
10         Save $\boldsymbol{f}_{\boldsymbol{l}_j}$ as a new cluster;
11         Compute the cluster characteristics for $\boldsymbol{f}_{\boldsymbol{l}_j}$: the number of lines, the
             average length, and the average azimuth;
12         Create an average line segment chosen as a representative line segment
             of $\boldsymbol{f}_{\boldsymbol{l}_j}$ and add it into $ALS$;
13     Delete all line segments belonging to $\boldsymbol{f}_{\boldsymbol{l}_j}$ from the dataset $L$;
14     $j \leftarrow j + 1$;
   end
15 Return $ALS$;
```

Then all remaining line segments are sorted according to their length decreasingly (line 2). The main part of the algorithm processes them one by one in this order (line 4). In each round (lines 5-14), the longest non-processed line segment $\boldsymbol{l}_j$ is selected to form a cluster, and all near line segments are tried to be attached to this cluster (lines 6-7). To choose the line segments belonging to the currently formed cluster, a *buffer zone* is introduced as a maximally allowed space surrounding the selected line segment $\boldsymbol{l}_j$. In other words, the buffer zone determines a minimum distance between two line segments of adjacent clusters. This buffer zone decides which line segment is close enough to be inserted into the cluster. The size of the buffer zone *bfs* is computed as:

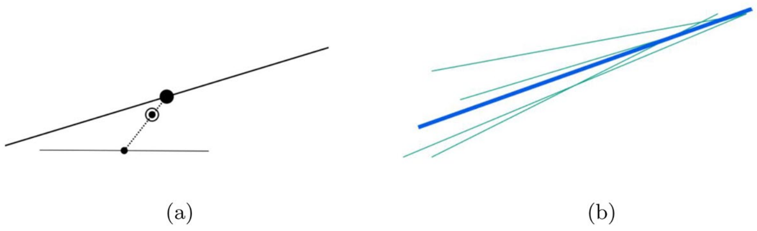


Fig. 2 Example of average line computation **a** construction of the cluster center for a cluster with two line segments, **b** example of the average line (bold) of the cluster with four lines (thin)

$$bfs = min(max(bfs_{\min}, \frac{1}{10}\lambda_j), bfs_{\max}) \tag{4}$$

where λ_j is the length of the current investigated line segment $\boldsymbol{l}_j$. In other words, the buffer zone becomes bigger according to the length of the line segment concerning the minimal $bfs_{\min}$ and the maximal $bfs_{\max}$ distance.

A line segment is inserted into the current cluster only if its azimuth is similar to the azimuth of the longest line segment in the cluster (the limits of similarity are given by the user parameter azimuth threshold ξ_{az}, line 8).

If the cluster formed this way contains enough line segments (the required number is given by ξ_{cnt}, line 9), it is saved (line 10), and the average line segment of this cluster is computed (lines 11-12) using three parameters: the center, the length and the azimuth. These parameters are determined from the cluster characteristics. The length and the azimuth are average values computed from all line segments in the cluster. The center is obtained as a length-weighted average of coordinates of each line segment center in the cluster (see Fig. 2). All the line segments forming the cluster are removed from the data set (line 13), and the algorithm continues with the next yet unprocessed line segment.

Let us move to the time complexity of Algorithm 1. The whole algorithm can be divided into three main parts.

Part 1 (line 1), line segments insertion into the grid and filtration, needs $\mathcal{O}_r(N)$ time supposing maximally $\mathcal{O}(\sqrt{N}) \times \mathcal{O}(\sqrt{N})$ size of the grid.

For part 2 (line 2), sorting, some well-known sorting algorithms such as quicksort or merge sort can be used. Using an optimal sorting algorithm, part 2 needs $\mathcal{O}_s(N\log N)$ time.

The worst scenario of part 3 (line 3-14), the clustering part, can lead to a situation where all checked line segments in L are outside the buffer zones, which means that in every iteration a new cluster is made. As a consequence, step 7 will check all line segments in $\mathcal{O}(N)$. The overall time complexity of the clustering part is thus $\mathcal{O}_{cl}(N^2)$.

The total time complexity of clustering line analysis of the MHHC can be summarized as $\mathcal{O}_{MHHC} = \mathcal{O}_r + \mathcal{O}_s + \mathcal{O}_{cl} = \mathcal{O}(N) + \mathcal{O}(N\log N) + \mathcal{O}(N^2) = \mathcal{O}(N^2)$.

Algorithm 1 checks many possible cases of cluster groups and brings a usable solution. However, due to the extensive search, it is very slow. A more sophisticated clustering method can improve the time complexity and substantially speed up the computation.

5 The proposed method for line segments clustering

The original local search algorithm expects all the point coordinates of the same units in the same interval to work correctly. However, suppose the line segments given by Eq. (3) are used instead of points. In that case, the particular coordinates can have different intervals and units (the azimuth is measured in degrees, and the spatial coordinates are usually in meters).

To balance such heterogeneous coordinates, let us define the *weight* $\mathbf{w}$ for distance computations. The calculation of the *weighted distance* d_w of two points $\mathbf{p} = (p_1, p_2, \dots p_D)^T$ and $\mathbf{q} = (q_1, q_2, \dots q_D)^T$ affected by the weight $\mathbf{w} = (w_1, w_2, \dots w_D)^T, w_i \in R^+_\emptyset$ is defined as follows:

$$d_w(\mathbf{p}, \mathbf{q}) = \sqrt{\mathbf{w}\,(\mathbf{p} - \mathbf{q})^2} \tag{5}$$

The domain for $\mathbf{w}$ depends on the application or user preferences. By the weights, adding more influence to some of the coordinates is possible.

The final clustering cost Q in Eq. (1) depends on the individual coordinates' weight changes. Let us demonstrate the influence for $D = 2$. If the weight of the second coordinate is lower than the weight of the first coordinate, then the final clusters are more elongated in the direction of the second coordinate; see Fig. 3.

If the weights are used, the overall clustering cost Q is influenced by changing the weighted distance calculation individually d_w. Before calculating the weighted distance $d_w(\boldsymbol{l}_p, \boldsymbol{l}_q)$ between line segments $\boldsymbol{l}_p$ and $\boldsymbol{l}_q$ according to Eq. (5), line segments are tested if they lie in the buffer zone and satisfy the azimuth threshold ξ_{az}. If a segment violates one of these thresholds, the weighted distance d_w is set to infinity. Otherwise, the computation continues according to Eq. (5).

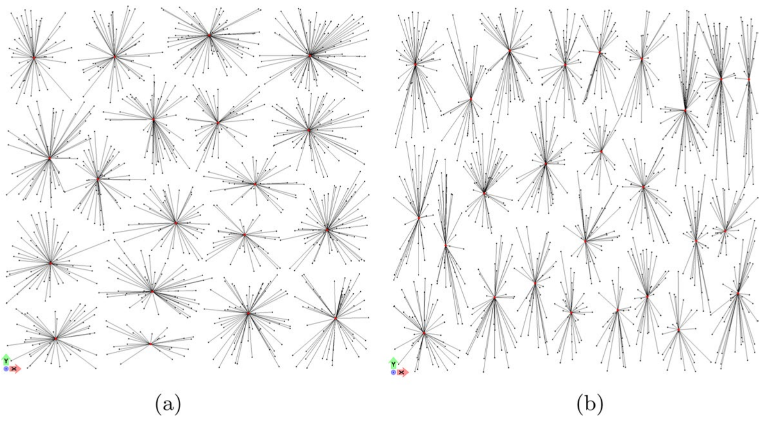


Fig. 3 Influence of the weights, **a** equal weights, **b** the weight of the second coordinate is smaller

So, the computation of the weighted distance d_w of two line segments $\boldsymbol{l}_p$ and $\boldsymbol{l}_q$ can be formalized as:

$$d_w(\boldsymbol{l}_p, \boldsymbol{l}_q) = \begin{cases} \sqrt{\boldsymbol{w} * (\boldsymbol{p} - \boldsymbol{q})^2} & \text{if } \boldsymbol{p}, \boldsymbol{q} \text{ fulfil the buffer zone condition and} \\ & \text{azimuth threshold } \xi_{az} \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

The proposed algorithm uses the distance d_w in Eq. (1) instead of d. This modification makes a connection between two line segments violating one of the conditions, which is costly but not impossible due to the randomness of the local search step. It means that, unlike the original cluster line analysis algorithm in MHHC presented in Sect. 4, the resulting cluster may violate either the buffer zone or the azimuth condition.

Another impact of Eq. (6) can be seen on special occasions, where multiple line segments are in the same buffer zone but could represent different lineament features. In that case, the distance between the cluster and line segments is calculated to determine whether the line segment should be connected to the existing cluster or creates a new cluster according to the total clustering cost Q.

The newly proposed algorithm is presented in Algorithm 3, using facility location for input data given as a set of line segments in Algorithm 2.

Algorithm 2: Facility location clustering

```
   Data: Array of line segments $L = \{\boldsymbol{l}_i, i = 1, 2, ..., N\}$, a distance function $d(\cdot, \cdot)$
   Result: $LC$ - list of clusters with their line segments.
1  Generate an initial clustering solution and save it into $LC$;
2  repeat $\mathcal{O}(NlogN)$ times
3      Pick $\boldsymbol{l}_i$ in $LC$ at random;
4      if $gain(\boldsymbol{l}_i) > 0$ then
           // a new facility to be opened
5          Perform reassignments and closures (modify $\boldsymbol{l}_i$ connections in $LC$);
6  Return $LC$ with the corresponding line segments;
```

The main differences against MHHC are as follows. The cluster line analysis in the MHHC algorithm uses filtering (noise reduction) as the first step to save computation time. In the proposed algorithm, it is no more needed. It is advantageous because filtering may expel some "useful" line segments and requires extra time. Instead of being sorted, the line segments are clustered using the facility location.

Another difference from the clustering algorithm in MHHC is a simplification of the buffer zone's shape around the cluster's average line segment. Now a rectangular buffer zone is used, and its shape is checked whenever the arrangement of the cluster is changed (new line segments are added or removed from the cluster) and modified if necessary. The width and the height of the buffer zone rectangle are controlled by the parameters X and Y, respectively.

Algorithm 3: AMHHC - cluster analysis

```
Data: Array of line segments $L = \boldsymbol{l}_i, i = 0, 1, ..., N-1$; the rectangular buffer
      zone width $X$ and height $Y$; the azimuth threshold $\xi_{az}$; the count
      threshold $\xi_{cnt}$; distance function, weights $w_i$
Result: $ALC$ - list of representative (average) line segments.
1 List of clusters $LC$ with their clients $\leftarrow$ FacilityLocation($L$);
2 foreach cluster $\boldsymbol{f} \in LC$ do
3     if the number of line segments in $\boldsymbol{f} > \xi_{cnt}$ then
4         Save $\boldsymbol{f}$ as a new cluster;
5         Compute the cluster characteristics for $\boldsymbol{f}$: the number of lines, the
            average length, and the average azimuth;
6         Create an average line segment which is chosen as a representative line
            segment of $\boldsymbol{f}$ and add it into $ALS$;
  end
7 Return $ALS$;
```

Let us address the complexity of AMHHC. The worst-case complexity of the proposed algorithm is reigned by the facility location part (step 1 in Algorithm 2), which is $\mathcal{O}(N^2)$. It means that the AMHHC algorithm is, in the worst case, not better than the MHHC; however, the expected behavior could be better due to the more sophisticated clustering part.

6 Experiments and results

The newly proposed AMHHC algorithm and the existing MHHC algorithm were implemented in Python 3.7 (Kaas 2017) to make a fair comparison. Experiments were run on the computer with Intel ® Xeon ® under Windows 10 with 14 GB memory.

The set of parameters for the original MHHC algorithm was chosen to produce the best possible MHHC results: $\xi_{\mathrm{cnt}} = 4$, $\xi_{az} = 20°$, $bfs_{\max} = 200m$ and $bfs_{\min} = 100m$. The parameters for AMHHC were then chosen similarly for a fair comparison. The parameters for AMHHC were: $\xi_{\mathrm{cnt}} = 4$, $\xi_{az} = 20°$, $X = 150m$ and $Y = 200m$.

The datasets for the tests were not taken from one particular geographical area but combined from more regions; the main criterion was the number of detected lineaments so that we could produce a table with a growing number of lineaments and compare the runtimes as a function of input size. The tests were run on 13 datasets of various sizes, 1000 up to 300 000 line segments (Kaas and Silhavy 2017a, b). The datasets describe natural areas in Canada, the Czech Republic (the Šumava mountains), and the Slovak Republic (the Žiar and Hronska mountains). Some of these areas were visualized in Silhavy et al. (2016). The two most extensive datasets could not have been processed by the MHHC method due to their more considerable memory requirements.

The quality of results was compared in two ways, by mutual correlations computed from the results and by visual comparison.

The correlation of the line segments was computed by the vector-based comparison (Silhavy et al. 2016), see Algorithm 4. This algorithm has two sets of line segments as input, A, and B. It takes a line segment from A and looks for a corresponding line segment in B. A corresponding line segment must lie in the defined buffer zone and have the azimuth difference from the tested line segment less than ξ_{az}. The correlation is then determined as the number of the found reference line segments and the tested line segments in percent. A bigger value means a bigger correspondence between both line segment sets. The same test is done with swapped A and B sets.

```
Algorithm 4: Corr($L_A$, $L_B$)
  Data: Sets of line segments $L_A = \{\boldsymbol{l}_{Ai}\}, L_B = \{\boldsymbol{l}_{Bj}\}$ where $i = 0, 1, \ldots, |L_A|$
        and $j = 0, 1, \ldots |L_B|$; the rectangular buffer zone width $X$ and height
        $Y$; the azimuth threshold $\xi_{az}$
  Result: Correlation between sets $L_A$ and $L_B$ [%]
1 counter ← 0 ;
2 foreach $\boldsymbol{l}_{Ai} \in L_A$ do
3     Make a buffer zone about $\boldsymbol{l}_{Ai}$ using Eq. 4;
4     if line segment exists completely withing the buffer zone $L_B$
      and $(\|\boldsymbol{l}_{Ai} - \boldsymbol{l}_{Bj}\|_\alpha < \xi_{az})$ then
5         counter ← counter + 1;
  end
6 Return $(|L_A|/counter) * 100$;
```

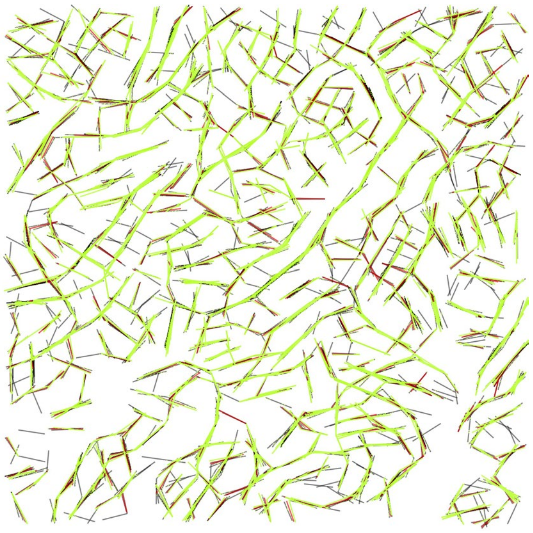
Table 1, column Corr(MHHC, AMHHC), uses the MHHC line segments as L_A and the AMHHC line segments as L_B, Corr(AMHHC, MHHC) vice versa. Mutual correlations proved a high rate of identity between the results of both methods. Corr(MHHC, AMHHC) reached nearly 100 percent, which means that nearly for

Table 1 Correlation and count of clusters comparison

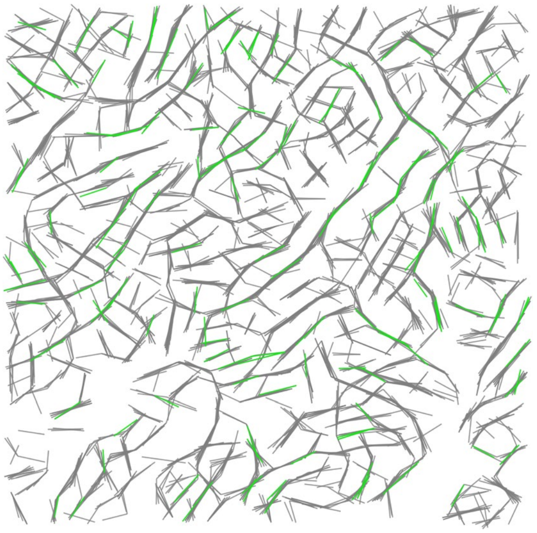
#Lsg	Corr(MHHC, AMHHC)	Corr(AMHHC, MHHC)	Corr(MHHC$_{g-}$, AMHHC)	Corr(AMHHC, MHHC$_{g-}$)	#cls MHHC	#cls MHHC $_{g-}$	#cls AMHHC
1 000	97.1	98.6	98.6	99.0	88	92	384
2 000	99.7	98.9	99.7	99.3	144	145	570
3 500	99.6	92.7	99.9	96.4	294	321	927
5 000	99.3	94.4	99.5	97.6	458	475	985
10 000	99.1	93.2	99.7	98.8	851	880	1624
15 000	99.0	90.7	99.8	97.6	1268	1372	2812
20 000	99.4	95.7	99.5	98.2	1854	1870	3321
25 000	99.3	96.8	99.8	98.0	2300	2373	4830
35 000	99.5	94.8	99.9	98.3	3010	3061	5966
50 000	99.6	94.5	99.9	97.5	3875	4158	6235
60 000	99.0	95.2	99.6	99.6	4769	4822	9214

#Lsg Number of line segments, multihillshade hierarchic clustering—MHHC, accelerated multihillshade hierarchic clustering—AMHHC, the MHHC without grid filtration—MHHC$_{g-}$, #Cls number of clusters

Fig. 4 Visual comparison of results (Note multihillshade hierarchic clustering—MHHC, accelerated multihillshade hierarchic clustering—AMHHC)



(a) Input dataset (gray), MHHC (red) and AMHHC (light green) centers.



(b) Input dataset (gray) and AMHHC centers (green) not present in MHHC.

each line segment computed by MHHC there is a corresponding line segment computed by AMHHC. Corr(AMHHC, MHHC) came to 95 percent on average, implying that some line segments are computed by AMHHC that do not have a corresponding line segment found by MHHC. These extra line segments in AMHHC are usually those which have been filtered in the noise reduction step in MHHC.

Table 2 Runtime of multihillshade hierarchic clustering (MHHC), accelerated multihillshade hierarchic clustering (AMHHC), and the speed-up

#Lsg	MHHC [s]	AMHHC [s]	MHHC/AMHHC
1 000	179	81	2.21
2 000	459	179	2.56
3 500	935	300	3.12
5 000	2553	856	2.98
10 000	9122	2425	3.76
15 000	21465	5422	3.96
20 000	30578	9173	3.33
25 000	63360	16439	3.85
35 000	68565	19589	3.50
50 000	157583	36636	4.30
60 000	140877	31084	4.53

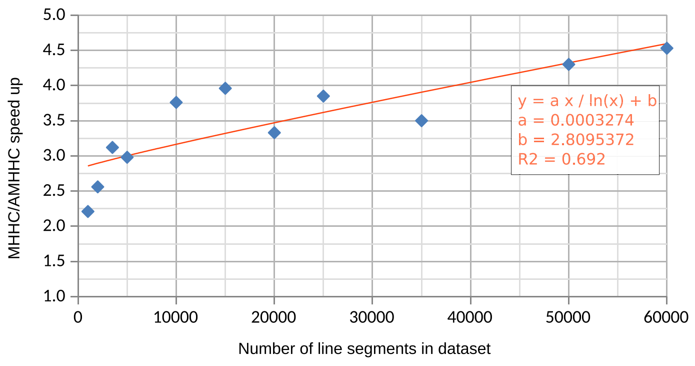


Fig. 5 Accelerated multihillshade hierarchic clustering (AMHHC) speed-up against the original multihillshade hierarchic clustering (MHHC)

Therefore the $MHHC_{g\_}$ represents the MHHC algorithm without grid filtration (step 1 in Algorithm 1). This phenomenon is confirmed by column Corr(AMHHC, $MHHC_{g-}$), which has increased the correlation to 98 percent on average. The last three columns show the number of resulting clusters in all methods. In summary, we can say that the AMHHC provides similar results (reaching 98 to 100 percent of correlation) to the original MHHC approach.

The statistical results were confirmed by visual comparison. A typical example is shown in Fig. 4a, where the input dataset is represented by gray color, average centers of MHHC by red, and AMHHC by light green. It can be seen that the results are nearly the same. The differences are minimal; see Fig. 4b, where greenly highlighted average centers produced by AMHHC are not present in the result of MHHC. These extra centers are usually present in regions with a higher density of line segments. AMHHC is in some way more careful.

Table 2 shows the total computing times of the original MHHC and AMHHC. The speed-up of AMHHC is shown in the column “MHHC/AMHHC”.

It can be seen that AMHHC is for tested datasets faster; the trend is shown in Fig. 5, i.e., for bigger datasets, there comes a higher speed-up between the proposed algorithm and its predecessor. The speed-up is caused by a more sophisticated clustering approach. The first part of the Algorithm 2 continuously investigates a point against existing facilities. In real data, the number of facilities slowly increases during the algorithm. Therefore, the $\mathcal{O}(N^2)$ time is rarely needed in practice. The second part of the Algorithm 2 with complexity $\mathcal{O}(N\log N)$ is enough to make a sufficient clustering solution. In contrast, in MHHC a point is investigated against the rest of the data set; therefore, $\mathcal{O}(N^2)$ is always needed.

7 Conclusion

This paper presented a new algorithm, AMHHC, for the spatial line segment clustering in the lineament extraction. The algorithm is based on a modification of the MHHC algorithm. The modification is based on the facility location algorithm for clustering, modified for line segments. The experiments verified that the algorithm is much faster than the original algorithm and provides nearly identical results, keeping the good properties of the original algorithm. Its minor memory requirements allow it to process more extensive data sets.

The proposed solution could be further accelerated using the N-dimensional version of quadtree to restrict the searched space of line segments in the function *gain* used in Algorithm 2. Instead of checking the whole input data set of the line segments, only those line segments, which can be attached to the newly formed cluster, would be inspected.

Acknowledgements This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic—project LO1506, UWB project SGS-2016-013 Advanced Graphical and Computing Systems, UWB project SGS-2016-004 Application of mathematics and informatics in geomatics and by the University specific research—1311. The authors would like to thank Mr. Maňák for his help with the processing of the results.

Funding Open access publishing supported by the National Technical Library in Prague.

References

Abdullah A, Akhir JM, Abdullah I (2010) Automatic mapping of lineaments using shaded relief images derived from digital elevation model (DEMs) in the Maran-Sungi Lembing area, Malaysia. Electron J Geotechn Eng 15(6):949–958

Adiri Z et al (2017) Comparison of Landsat-8, ASTER and Sentinel 1 satellite remote sensing data in automatic lineaments extraction: A case study of Sidi Flah-Bouskour inlier, Moroccan Anti Atlas. Adv Space Res 60(11):2355–2367

Banerjee A et al (2005) Clustering with Bregman divergences. J Mach Learn Res 6:1705–1749

Baraldi A, Blonda P (1999) A survey of fuzzy clustering algorithms for pattern recognition. I. IEEE Trans Syst Man Cybern Part B (Cybern) 29(6):778–785

Beckmann N et al (1990) The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of the 1990 ACM SIGMOD international conference on management of data, pp 322–331

Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517

Bonetto S et al (2015) A tool for semi-automatic linear feature detection based on DTM. Comput Geosci 75:1–12

Charikar M, Guha S (1999) Improved combinatorial algorithms for the facility location and k-median problems. In: 40th annual symposium on foundations of computer science (Cat. No. 99CB37039). IEEE, pp 378–388

Dubes R, Jain AK (1980) Clustering methodologies in exploratory data analysis. Adv Comput. Elsevier, Netherlands, pp 113–228

Ester M et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. KDD-96 Proceedings 96(34):226–231

Hashim M et al (2013) Automatic lineament extraction in a heavily vegetated region using Landsat Enhanced Thematic Mapper (ETM+) imagery. Adv Space Res 51(5):874–890

Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice-Hall Inc., USA

Jain AK, Flynn PJ (1996) Image segmentation using clustering. IEEE Press, Piscataway

Jordan G, Schott B (2005) Application of wavelet analysis to the study of spatial pattern of morphotectonic lineaments in digital terrain models. A case study. Remote Sens Environ 94(1):31–38

Kaas O (2017) MHHCA-Python. v1.1. 2017. https://github.com/OKaas/LineamentExtraction-MHHCA-Python

Kaas O, Silhavy J (2017a) MHHC-Input, v1.0. 2017. https://github.com/OKaas/LineamentExtraction-MHHC-Input

Kaas O, Silhavy J (2017b) MHHC-Input, v1.0. 2017. https://github.com/OKaas/LineamentExtraction-MHHCA-Input

Kashima H et al (2008) K-means clustering of proportional data using L1 distance. In: 2008 19th international conference on pattern recognition. IEEE, pp 1–4

Karimi B, Karimi HA (2017) An automated method for the detection of topographic patterns at tectonic boundaries. In: The 9th international conferences on pervasive patterns and applications. pp 72–77

Korupolu MR, Plaxton CG, Rajaraman R (1998) Analysis of a local search heuristic for facility location problems. In: SODA, vol 98. Citeseer, pp 1–10

Linde Y, Buzo A, Gray R (1980) An algorithm for vector quantizer design. IEEE Trans Commun 28(1):84–95

Lee JG, Han J, Whang KY (2007) Trajectory clustering: a partition-and-group framework. In: Proceedings of the 2007 ACM SIGMOD international conference on management of data. pp 593–604

Legendre P, Legendre LFJ (2012) Numerical Ecology. Elsevier

Likas A, Vlassis N, Verbeek J (2003) The global k-means clustering algorithm. Pattern Recogn 36(2):451–461

MacQueen J et al (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability. Vol. 1. 14. Oakland, CA, USA, pp 281–297

Mallast U et al (2011) Derivation of groundwater ow-paths based on semi-automatic extraction of lineaments from remote sensing data. Hydrol Earth Syst Sci 15(8):2665

Mohammadpour M, Bahroudi A, Abedi M (2020) Automatic lineament extraction method in mineral exploration using CANNY algorithm and hough transform. Geotectonics 54(3):366–382

Meyerson A (2001) Online facility location. In: Proceedings 42nd IEEE symposium on foundations of computer science. IEEE, pp 426–431
Mao J, Jain AK (1996) A self-organizing network for hyperellipsoidal clustering (HEC). IEEE Trans Neural Netw 7(1):16–29
Masoud AA, Koike K (2011) Auto-detection and integration of tectonically significant lineaments from SRTM DEM and remotely-sensed geophysical data. ISPRS J Photogramm Remote Sens 66(6):818–832
Masoud A, Koike K (2017) Applicability of computer-aided comprehensive tool (LINDA: LINeament Detection and Analysis) and shaded digital elevation model for characterizing and interpreting morphotectonic features from lineaments. Comput Geosci 106:89–100
Minár J, Sládek J (2009) Morphological network as an indicator of a morphotectonic field in the central Western Carpathians (Slovakia). Zeitschrift für Geomorphologie, Suppl. Issues 53(2):23–29
Msaddek MH et al (2019) Applicability of developed algorithm for semi-automated extraction and morphotectonic interpretation of lineaments using remotely sensed data, Southwestern Tunisia. Remote Sens Earth Syst Sci 2(4):292–307
Nyberg B, Nixon CW, Sanders DJ (2018) NetworkGT: a GIS tool for geometric and topological analysis of two-dimensional fracture networks. Geosphere 14(4):1618–1634
O'Leary DW, Friedman JD, Pohn HA (1976) Lineament, linear, lineation: some proposed new standards for old terms. Geol Soc Am Bull 87(10):1463–1469
Panagiotakis C, Kokinou E (2014) Linear pattern detection of geological faults via a topology and shape optimization method. IEEE J Select Topics Appl Earth Observ Remote Sens 8(1):3–11
Ramli MF et al (2010) Lineament mapping and its application in landslide hazard assessment: a review. Bull Eng Geol Env 69(2):215–233
Rasmussen EM (1992) Clustering algorithms. Inf Retr Data Struct Algorithms 419:442
Rahnama M, Gloaguen R (2014) TecLines: a MATLAB-based toolbox for tectonic lineament analysis from satellite images and DEMs, Part 1: Line segment detection and extraction. Remote Sensing 6(7):5938–5958
Russell S, Lodwick W (1999) Fuzzy clustering in data mining for telco database marketing campaigns. In: 18th international conference of the North American fuzzy information processing society-NAFIPS (Cat. No. 99TH8397). IEEE, pp 720–726
Salui CL (2018) Methodological validation for automated lineament extraction by LINE method in PCI Geomatica and MATLAB based Hough transformation. J Geol Soc India 92(3):321–328
Soto-Pinto C, Arellano-Baeza A, Sánchez G (2013) A new code for automatic detection and analysis of the lineament patterns for geophysical and geological purposes (ADALGEO). Comput Geosci 57:93–103
Seleem TA (2013) Analysis and tectonic implication of DEM-derived structural lineaments, Sinai Peninsula, Egypt. Int J Geosci 4(1):41016. https://doi.org/10.4236/ijg.2013.41016
Silhavy J et al (2016) A new artefacts resistant method for automatic lineament extraction using Multi-Hillshade Hierarchic Clustering (MHHC). Comput Geosci 92:9–20
Skála J, Kolingerová I (2011) Dynamic hierarchical triangulation of a clustered data stream. Comput Geosci 37(8):1092–1101
Thiele ST et al (2017) Rapid, semi-automatic fracture and contact mapping for point clouds, images and geophysical data. Solid Earth 8(6):1241–1253
Vasuki Y et al (2017) An interactive image segmentation method for lithological boundary detection: a rapid mapping tool for geologists. Comput Geosci 100:27–40
Vaz DA (2011) Analysis of a Thaumasia Planum rift through automatic mapping and strain characterization of normal faults. Planet Space Sci 59(11–12):1210–1221
Xu J et al (2020) Automatic extraction of lineaments based on wavelet edge detection and aided tracking by hillshade. Adv Space Res 65(1):506–517
Yeomans CM et al (2019) Integrated object-based image analysis for semi-automated geological lineament detection in southwest England. Comput Geosci 123:137–148