RAD HAZU. MATEMATIČKE ZNANOSTI
Vol. 23 = 538 (2019): 123-140
DOI: https://doi.org/10.21857/yvjrdcqq8y

A FAST AND EFFICIENT METHOD FOR SOLVING THE MULTIPLE LINE DETECTION PROBLEM

Rudolf Scitovski, Una Radojičić and Kristian Sabo

Abstract. In this paper, we consider the multiple line detection problem on the basis of a data points set coming from a number of lines not known in advance. A new and efficient method is proposed, which is based upon center-based clustering, and it solves this problem quickly and precisely. The method has been tested on 100 randomly generated data sets. In comparison to the incremental algorithm, the method gives significantly better results. Also, in order to identify a partition with the most appropriate number of clusters, a new index has been proposed for the case of a cluster whose lines are cluster-centers. The index can also be generalized for other geometrical objects.

1. Introduction

Let $\mathcal{A} = \{a^i = (x_i, y_i) \colon i = 1, \dots, m\} \subset \Delta \subset \mathbb{R}^2$, $\Delta = [0, a] \times [0, b]$, be a data points set coming from $k \geq 2$ lines

$$\ell_j \equiv u_j x + v_j y + z_j = 0, \quad u_j^2 + v_j^2 \neq 0, \quad j = 1, \dots, k, \tag{1.1}$$

in the plane not known in advance that should be reconstructed or detected. Let us suppose that data coming from a line are homogeneously distributed around that line, such that random errors from normal distribution with expectation 0 are added to uniformly distributed points on the line in the direction of a normal.

The multiple line detection problem finds application in various fields, such as computer vision and image processing [5,11], robotics [22], laser range measurements [6], and crop row detection in agriculture [10,25], etc.

Many approaches to this problem can be found in the literature. The Hough transform is the most popular application [4,13]. This method assigns to every point $a = (x, y) \in \mathcal{A}$ a set of all lines passing through that point.

2010 *Mathematics Subject Classification.* 65K05, 90C26, 90C27, 90C56, 90C57, 05E05.
Key words and phrases. Location, multiple line detection problem, center-based clustering, number of clusters, globally optimal partition.

In this way, the point is represented by means of a set of lines in the Hesse normal form that are defined by a set of parameters:

$$\{(\alpha,\gamma)\in\mathbb{R}^2\colon x\cos\alpha + y\sin\alpha - \gamma = 0\}.$$

In other words, point $a\in\mathcal{A}$ is represented by a set of points $(\alpha,\gamma)\in\mathbb{R}^2$ that lie in the so-called Hough plane. The original points close to one of the lines intensify the points in the Hough plane. Various line detection algorithms are based on recognizing the most intensive points in the Hough plane. The main drawback is that algorithms based on Hough transforms are slow and various approaches have been used to overcome this problem. E.g., [5] presented an improved voting scheme for the Hough transform for real-time line detection. Line and circle detection based on Hough transforms is considered in [11].

The other group of methods is based on the center-based clustering approach, where one of the first papers is [19]. In relation thereto, [1] considers the clusterwise linear regression problem by using the incremental method (see also [2,18]), and a line detection algorithm based on probabilistic clustering is proposed in [3] and compared to the fuzzy clustering approach in applications to real-world images. [25] proposes a combination of center-based clustering and total least squares for the purpose of solving the crop row detection problem in agriculture.

The paper is organized as follows. In the next section, we define a general center-based clustering problem and narrow it down to the multiple line detection problem. We also briefly describe the main methods adapted to solve this problem, i.e. the k-means and the incremental algorithm and propose a construction of a random data set coming from a number of lines in the plane. After that, a new proposed method is described in Section 3. Furthermore, a new index is proposed for recognizing a partition with the most appropriate number of clusters whose cluster-centers are lines. The proposed method has been tested on 100 randomly generated data sets. Finally, some conclusions are given in Section 4.

2. The multiple line detection problem as a special center-based clustering problem

First, let us briefly define a general center-based clustering problem. Let $d\colon \mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}_+$, $\mathbb{R}_+=[0,+\infty\rangle$, $d(u,v)=(u-v)^T(u-v)=\|u-v\|_2^2$, be the least squares distance-like function (see e.g. [9,16]), and let $\mathcal{A}=\{a^i\in\mathbb{R}^n\colon i=1,\dots,m\}$ be a finite subset in $\mathbb{R}^n$. A partition Π of the set $\mathcal{A}$ is a collection of nonempty disjoint subsets (clusters) $\pi_1,\dots,\pi_k$, $(1\le k\le m)$ whose union is $\mathcal{A}$. A set of all such partitions will be denoted by $\mathcal{P}(\mathcal{A};k)$.

If to each cluster $\pi_j\in\Pi$ we associate its center c_j defined by

$$c_j := \operatorname*{argmin}_{x\in\operatorname{conv}(\mathcal{A})} \sum_{a^i\in\pi_j} d(x,a^i), \tag{2.1}$$

then a globally optimal k-partition can be defined as a solution to the following *global optimization problem* (GOP):

$$\operatorname*{argmin}_{\Pi\in\mathcal{P}(\mathcal{A};k)} \mathcal{F}(\Pi), \qquad \mathcal{F}(\Pi)=\sum_{j=1}^{k}\sum_{a^i\in\pi_j} d(c_j,a^i), \qquad c=(c_1,\dots,c_k). \tag{2.2}$$

Conversely, for a given set of points $c_1,\dots,c_k\in\mathbb{R}^n$, by applying the minimal distance principle, we can define the partition $\Pi=\{\pi(c_1),\dots,\pi(c_k)\}$ of the set $\mathcal{A}$ consisting of clusters

$$\pi_j := \pi(c_j)=\{a\in\mathcal{A} : d(c_j,a)\le d(c_s,a),\ \forall s=1,\dots,k\}, \qquad j=1,\dots,k,$$

where clusters π_j should be mutually disjoint and where one has to have in mind that every element of the set $\mathcal{A}$ occurs in one and only one cluster. Hence the problem of finding an optimal partition of the set $\mathcal{A}$ can be reduced to the following GOP (see e.g. [9, 15, 20]):

$$\operatorname*{argmin}_{c\in\operatorname{conv}(\mathcal{A})^k} F(c), \qquad F(c)=\sum_{i=1}^{m}\min_{1\le j\le k} d(c_j,a^i), \tag{2.3}$$

better known as the center-based clustering problem. The solutions of (2.2) and (2.3) coincide [18, 20].

The problem of recognizing k lines $\ell_1,\dots,\ell_k$ in the plane will be treated as a special center-based clustering problem; i.e. the set $\mathcal{A}$ will be divided into k-clusters whose centers will be lines $\ell_1,\dots,\ell_k$. Let us assume that points coming from any line are uniformly allocated in the neighborhood of that line.

Remark 1. Note that this practically means that our problem can also be treated as a special case of the multiple segment detection problem (see e.g. [21]), where border points of line segments lie on different sides of the rectangle Δ.

If the distance from the point $a^i\in\mathcal{A}$ to the line $\ell(u,v,z)\equiv ux+vy+z=0$, $u^2+v^2\ne 0$ is defined as:

$$\mathfrak{D}(\ell,a^i)=\frac{(ux_i+vy_i+z)^2}{u^2+v^2}, \tag{2.4}$$

then according to (2.3), the multiple line detection problem can be defined as the following GOP:

$$\operatorname*{argmin}_{\mathbf{u},\mathbf{v},\mathbf{z}\in\mathbb{R}^k} F(\mathbf{u},\mathbf{v},\mathbf{z}), \qquad F(\mathbf{u},\mathbf{v},\mathbf{z})=\sum_{i=1}^{m}\min_{1\le j\le k}\mathfrak{D}(\ell_j(u_j,v_j,z_j),a^i). \tag{2.5}$$

Note that, especially for $k=1$, problem (2.5) is reduced to searching for the best total least squares (TLS) line [14]:

$$\operatorname*{argmin}_{u,v,z\in\mathbb{R}} F(u,v,z), \qquad F(u,v,z)=\sum_{i=1}^{m}\mathfrak{D}(\ell(u,v,z),a^i). \tag{2.6}$$

The function F from (2.5) is a symmetric Lipschitz continuous, but non-convex and non-differentiable function and `GOP` (2.5) can have a large number of independent variables. Because of that, direct application of some global optimization method would not be efficient.

2.1. *Modification of the k-means algorithm.* The most popular and most frequently used method for solving `GOP` (2.5) is the well-known k-means algorithm. In the case when lines are cluster-centers, this algorithm will be modified in the following way:

ALGORITHM 1. (The k-closest line algorithm (`KCL`))

Step A: (Assignment step) For each set of mutually different lines $\ell_1, \dots, \ell_k$, the set $\mathcal{A}$ should be divided into k disjoint unempty clusters $\pi_1, \dots, \pi_k$ by using the minimal distance principle

$$\pi_j := \{a \in \mathcal{A} \colon \mathfrak{D}(\ell_j, a) \le \mathfrak{D}(\ell_s, a),\ \forall s \ne j\}, \quad j = 1, \dots, k; \tag{2.7}$$

Step B: (Update step) Given a partition $\Pi = \{\pi_1, \dots, \pi_k\}$ of the set $\mathcal{A}$, one can define the corresponding line-centers $\hat{\ell}_1, \dots, \hat{\ell}_k$ by solving `GOP`'s

$$\operatorname*{argmin}_{u,v,z \in \mathbb{R}} f_j(u, v, z), \quad f_j(u, v, z) = \sum_{a \in \pi_j} \mathfrak{D}(\ell_j(u, v, z), a), \quad j = 1, \dots, k. \tag{2.8}$$

If we are able to find a good enough approximation of line-centers, then the `KCL` algorithm is run with `Step A`, and if we are able to find a good enough initial partition, then the `KCL` algorithm is run with `Step B`.

Similarly to the case of ordinary cluster-centers, it can be seen that the sequence of the objective function values $F^{(n)} = F(\mathbf{u}^{(n)}, \mathbf{v}^{(n)}, \mathbf{z}^{(n)})$ decreases monotonically [18, 20]. Therefore the algorithm can be stopped when the following condition is met for some small $\epsilon_B > 0$ (say .005) (see [2]):

$$\tfrac{F^{(n-1)} - F^{(n)}}{F^{(n-1)}} < \epsilon_B. \tag{2.9}$$

Searching for the line which represents the cluster π_j in Step B is solved efficiently according to [14]. First, we determine a covariance matrix:

$$\Sigma_j = \tfrac{1}{|\pi_j|} \sum_{a^i \in \pi_j} (c_j - a^i)(c_j - a^i)^T, \tag{2.10}$$

where c_j is the centroid of the cluster π_j. The line searched for is a TLS-line $(u_0, v_0)^T((x, y) - c_j) = 0$, where (u_0, v_0) is a unit eigenvector corresponding to a smaller eigenvalue of the matrix Σ_j.

For solving `GOP` (2.5) by means of the `KCL` algorithm we will need a very good initial approximation $\hat{\ell}_1, \dots, \hat{\ell}_k$, i.e. as close to the solution as possible.

2.2. *Modification of the incremental method.* For solving multiple line detection problem (2.5), we can find a modification of the well-known incremental algorithm in the literature [1, 2, 18, 25] known as the *Incremental Method for Line Detection* (IMLD).

The algorithm is run by some initial center-line $\tilde{\ell}_1$. This can be a randomly selected line, but the best TLS-line is used more frequently [14, 25]. The next line $\tilde{\ell}_2$ will be obtained by using the DIRECT algorithm for the line in the Hesse normal form $\ell(\alpha,\gamma) \equiv x\cos\alpha + y\sin\alpha - \gamma = 0$. Note that in this case it is not acceptable to search for a line in implicit form as it is used in formula (2.4) since the DIRECT algorithm necessitates determining the finite domain of the objective function. So, in this case one uses the Hesse normal form for the next line.

Algorithm 2 (The incremental algorithm for a line in the Hesse normal form)

```
Input: $\mathcal{A} = \{a^i = (x_i, y_i) \colon i = 1,\dots,m\} \subset \Delta \subset \mathbb{R}^2$, $\Delta = [0,a] \times [0,b]$; $\epsilon > 0$;
1: Set $r = 2$ and $\mathtt{GO}_{min} = +\infty$;
2: According to [14], determine TLS-line $\tilde{\ell}_1(\tilde{u}_1, \tilde{v}_1, \tilde{z}_1)$ and calculate $F_1 = \sum_{i=1}^{m} \mathfrak{D}(\tilde{\ell}_1(\tilde{u}_1, \tilde{v}_1, \tilde{z}_1), a^i)$;
3: By using the DIRECT algorithm determine
     $$(\tilde{\alpha}, \tilde{\gamma}) \in \operatorname*{argmin}_{\substack{\alpha\in[0,2\pi],\\ \gamma\in\left[0,\sqrt{a^2+b^2}\right]}} G_r(\alpha,\gamma),$$
   (2.11)
     $$G_r(\alpha,\gamma) = \sum_{i=1}^{m} \min\{\delta_{r-1}^{(i)}, \mathfrak{D}_H(\ell(\alpha,\gamma), a^i)\},$$
   where $\mathfrak{D}_H(\ell(\alpha,\gamma), a^i) = (x_i\cos\alpha + y_i\sin\alpha - \gamma)^2$ and $\delta_{r-1}^{(i)} = \min\{\mathfrak{D}(\tilde{\ell}_1, a^i), \dots, \mathfrak{D}(\tilde{\ell}_{r-1}, a^i)\}$;
   Define $\tilde{\ell}_r(\cos\tilde{\alpha}, \sin\tilde{\alpha}, -\tilde{\gamma})$ and set $F_r = G_r(\tilde{\alpha}, \tilde{\gamma})$;
4: To the lines $\tilde{\ell}_1, \dots, \tilde{\ell}_r$ apply the KCL algorithm and denote the obtained partition by
   $\hat{\Pi}_r = \{\hat{\pi}_1(\hat{\ell}_1), \dots, \hat{\pi}_r(\hat{\ell}_r)\}$;
5: Calculate GO-index value GO(r) according to (3.3);
6: if $\mathtt{GO}(r) < \mathtt{GO}_{min}$, then set $\mathtt{GO}_{min} = \mathtt{GO}(r)$ and $\Pi^\star := \hat{\Pi}_r$ end if;
7: if $\frac{F_{r-1}-F_r}{F_1} < \epsilon$, then STOP; else set $\tilde{\ell}_1 = \hat{\ell}_1, \dots \tilde{\ell}_r = \hat{\ell}_r$ and $r := r+1$ and go to Step 3 end if;
Output: $\Pi^\star$.
```

According to [2], the iterative process stops when a relative objective function value is less than ϵ, and, by applying the `GO`-index defined in Subsection 3.2.1, we obtain the partition with the most appropriate number of clusters.

Problem (2.11) is most often a nonlinear and non-convex `GOP` that can have many points of local or global minima, but it can be successfully solved by using the well-known `DIRECT` algorithm for global optimization [7,8], where the line $\hat{\ell}_r$ is searched for in the Hesse normal form, where $\alpha \in [0, 2\pi]$ and $\gamma \in [0, \sqrt{a^2 + b^2}]$.

Example 2.1. We will consider a line detection problem for a set of data points $\mathcal{A}$ defined as in the next subsection. The initial center line $\tilde{\ell}_1$ will be defined as a TLS-line (Fig. 1a). `ContourPlot` of the minimizing function from (2.11) for $r = 2$ points to several points of the local minimum (see Fig. 1b). By using the `DIRECT` algorithm we obtain the next center-line $\tilde{\ell}_2$ and by using the `KCL` algorithm we obtain lines $\hat{\ell}_1, \hat{\ell}_2$ (see Fig. 1c). Finally, after four iterations, we obtain the solution (see Fig. 1d).

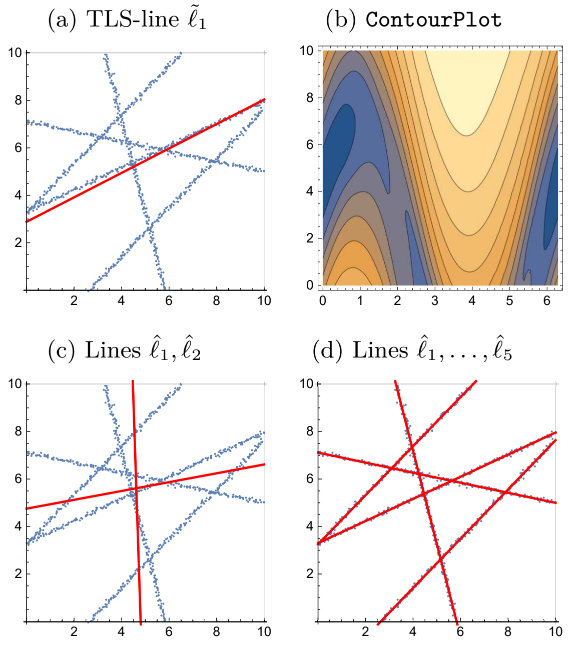

Figure 1. The `IMLD` algorithm

2.3. *Construction of a synthetic set of data points.* Let us assume that the points from $\mathcal{A}$ coming from any line $\ell_1, \dots, \ell_k$ are uniformly allocated in the neighborhood of part of that line visible in the rectangle Δ, and their number depends on the length of the line. Our method will be illustrated and tested on such sets of data. In this subsection, it will be shown how sets of such data can be generated randomly.

A set of data $\mathcal{A} \subset \Delta = [0, a] \times [0, b]$ coming from k lines whose graphs pass through a rectangle Δ will be defined in the following way. Let $h = 2$ and $N = 300$;

First, we choose a random integer $k \in \{2, 3, 4, 5\}$. After that, we randomly choose a few points A_j, B_j on the edge of the rectangle Δ not closer for h from the edges of the rectangle k times , such that they do not lie on the same side of the rectangle. Pairs of points $\{A_j, B_j\}$, $j = 1, \dots, k$ define k lines $\ell_1, \dots, \ell_k$ passing through the rectangle Δ (see Figure 2)

$$\ell_j(x, y) \equiv \begin{cases} x - A_j^1 = 0, & \text{if } A_j^1 = B_j^1, \\ (B_j^2 - A_j^2)x + (A_j^1 - B_j^1)y + A_j^2(B_j^1 - A_j^1) - A_j^1(B_j^2 - A_j^2) = 0, & \text{else.} \end{cases}$$

In order to have sets of data defined in the neighborhood of every line of approximately the same density, first we will define the number N corresponding to the number of points to be assigned to the diagonal of the rectangle Δ (a line with the greatest part visible in the rectangle Δ!). After that, on the line ℓ_j, we will choose $m_j = \lceil N\|A_j - B_j\|/\sqrt{a^2 + b^2}\rceil$ equidistantly allocated points $T_i = \lambda_i A_j + (1 - \lambda_i)B_j$, $i = 1, \dots, m_j$, where $\lambda_i = i/m_j$, and then to each point T_i we add a random error (ξ_i, η_i) from Bivariate Normal Distribution with expectation $0 \in \mathbb{R}^2$ and the covariance matrix $\sigma^2\mathbf{I}$, where $\sigma^2 = 0.05$. In this way, we have defined a cluster which belongs to the line ℓ_j

$$\pi_j = \{a^s \in \mathbb{R}^2 \colon a^s = T_s + (\xi_s, \eta_s),\ s = 1, \dots, m_j\}$$

and the set $\mathcal{A} = \pi_1 \cup \dots \cup \pi_k$ (see Figure 2). Note that the line ℓ_j does not to have to be the center of the cluster π_j. Four different situations with $2, 3, 4, 5$ lines are shown Fig 2. The efficiency of our method will be tested on such sets of data.

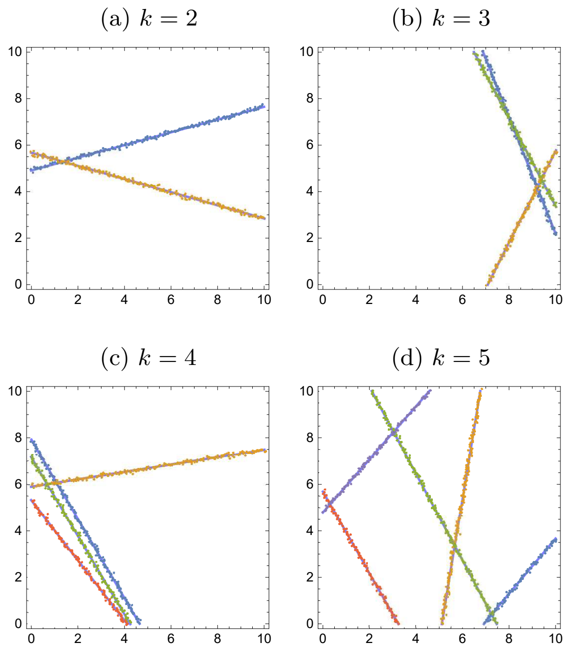

FIGURE 2. Four selected examples with $k = 2, 3, 4, 5$ lines

3. A new method

As already mentioned in Remark 1, due to the assumption of uniform allocation of points that, in the rectangle Δ, come from a line, the multiple line detection problem can also be treated as a special case of the multiple segment detection problem, where border points of line segments lie on different sides of the rectangle Δ. By using this assumption, we will construct a possible solution to problem (2.5), which will be called the *First Solution.* This solution will be used for searching for a globally optimal one.

3.1. *The First Solution.* The procedure of searching for the aforementioned First Solution will be shown in Algorithm 3, and algorithm steps will be illustrated by means of a simple example constructed as in Subsection 2.3, which is shown in Fig. 3a.

Algorithm 3 (`First solution`)

```
Input: $\mathcal{A} = \{a^i = (x_i, y_i) \colon i = 1, \dots, m\} \subset \Delta \subset \mathbb{R}^2$, $\Delta = [0,a] \times [0,b]$;
    $\epsilon > 0$;
1: Define the sets (see Fig. 3b):
       $\mathcal{B}_1 = \{x_i \in \mathcal{A} \colon |y_i| < \epsilon\}$ (bottom edge)
       $\mathcal{B}_2 = \{y_i \in \mathcal{A} \colon |a - x_i| < \epsilon\}$ (right edge)
       $\mathcal{B}_3 = \{x_i \in \mathcal{A} \colon |b - y_i| < \epsilon\}$ (upper edge)
       $\mathcal{B}_4 = \{y_i \in \mathcal{A} \colon |x_i| < \epsilon\}$ (left edge)
2: For every set $\mathcal{B}_j \neq \emptyset$ define a set of centers $B_j$ as a partition with the
    most acceptable number of clusters in $\mathcal{B}_j$ (see Fig. 3c);
3: Define a set of lines $\mathcal{P} = \{\ell_j \colon j = 1, \dots, K\}$, whose graphs are visible
    in the rectangle $\Delta$ and intersect two different sides of $\Delta$ at points from
    $B_1 \cup B_2 \cup B_3 \cup B_4$ (see Fig. 4a);
4: To the set $\mathcal{A}$ apply the minimum distance principle for lines from $\mathcal{P}$.
    Denote the obtained clusters by $\pi_1, \dots, \pi_K$;
5: For every pair $(\pi_j, \ell_j)$ define a new line $\hat{\ell}_j$ as the best TLS-line of the
    cluster $\pi_j$ (see Fig. 4b);
6: To every line $\hat{\ell}_j \in \{\hat{\ell}_j \in \mathcal{P} \colon |\pi_j| > 2\}$ assign the corresponding density
    $\rho_j = \frac{|\pi_j|}{|\hat{\ell}_j|}$. After that, group the lines by their densities into two groups.
    Let $\hat{\ell}_1, \dots, \hat{\ell}_r$ be a set of lines corresponding to a high-density cluster;
7: To the set $\hat{\ell}_1, \dots, \hat{\ell}_r$ apply the KCL algorithm and denote the obtained
    lines by $\ell_1^\star, \dots, \ell_r^\star$, and the corresponding clusters by $\pi_1^\star, \dots, \pi_r^\star$ (Fig. 4c);
Output: $\{(\ell_j^\star, \pi_j^\star),\ j = 1, \dots, r\}$.
```

Let us briefly describe Algorithm 3. For given $\epsilon > 0$, we consider elements from the set $\mathcal{A}$, which lie up to ϵ close to the edge of the rectangle Δ (Fig. 3b). Projection of these elements onto the corresponding bottom, right, upper or left edge of the rectangle Δ gives sets $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ from `Step 1`. These sets can be formed by a single loop through the set $\mathcal{A}$.

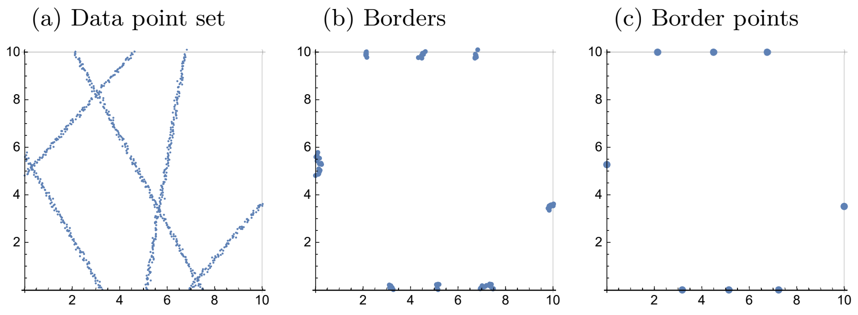

FIGURE 3. Construction of border points

In Step 2, optimal k-partitions ($k = 1, \dots, k_j$) are defined for every set $\mathcal{B}_j \neq \emptyset$ by means of the SymDIRECT global optimization algorithm (see [7,17]), where an 1-optimal partition is defined as the arithmetical mean of the set $\mathcal{B}_j$. The number k_j is determined by using the Davies-Bouldin (DB) index [23,24], where for DB(1) we take the variance of the set B_j. In this way, we determine sets of border points B_1, B_2, B_3, B_4 (Fig. 3c).

Every line from the set of lines $\mathcal{P} = \{\ell_j \colon j = 1, \dots, K\}$, defined in Step 3, is defined such that it intersects the rectangle Δ at points from different sets B_j (Fig. 4a). Step 3 is performed such that first the set $B = B_1 \cup B_2 \cup B_3 \cup B_4$ is defined and after that a set of pairs of points given below is found:

$$\{(IP_j, EP_j) \colon IP_j = (u_j, v_j),\ EP_j = (z_j, t_j),\ u_j \neq z_j \,\&\, v_j \neq t_j\} \subset B \times B,$$

through which lines from $\mathcal{P}$ pass.

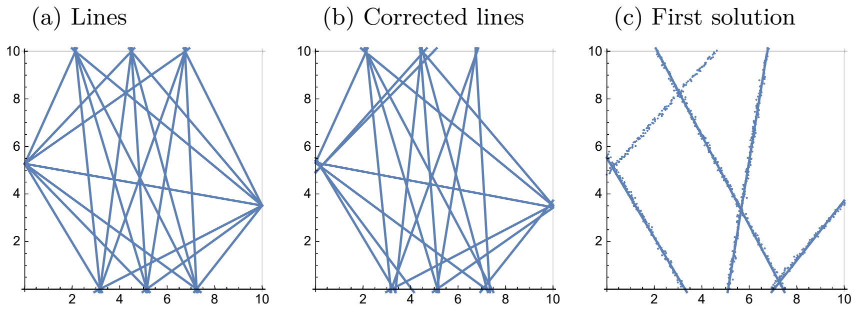

FIGURE 4. Searching for the First Solution

A corresponding cluster π_j is associated to every line $\ell_j \in \mathcal{P}$ in Step 4 by the minimal distance principle,

$$\pi_j = \{a^i \in \mathcal{A} \colon \mathfrak{D}(\ell_j, a^i) \leq \mathfrak{D}(\ell_s, a^i),\ \forall s = 1, \dots, K,\ s \neq j\}, \quad j = 1, \dots, K,$$

where the distance $\mathfrak{D}(\ell_j, a^i)$ from the point $a^i = (x_i, y_i) \in \mathcal{A}$ to the line $\ell_j : u_j x + v_j y + z_j = 0$ from $\mathcal{P}$ is given by (2.4). In this process, it can happen that some cluster is empty or that it has only a few elements. Such line is definitely not a candidate for the line we are looking for. Therefore, from the set $\mathcal{P}$, we leave out all lines ℓ_j for which the corresponding cluster π_j has less than three data from $\mathcal{A}$. In Step 5, the line $\hat{\ell}_j$ is associated to all other clusters π_j, ($|\pi_j| > 2$) as the best TLS-line [14] (Fig. 4b). Note that in this way we actually performed one iteration of the KCL algorithm.

In Step 6, from the set of remaining lines we will extract $r \leq K$ lines in whose neighborhood we find a significant number of data points. This will be done such that for every pair $(\hat{\ell}_j, \pi_j)$ we first define the density $\rho_j = \frac{|\pi_j|}{|\hat{\ell}_j|}$, where $|\hat{\ell}_j|$ is the length of part of the line $\hat{\ell}_j$ visible in the rectangle Δ. It can be expected that the density assigned to the lines searched for is greater than the density assigned to other lines. Thus a sequence of densities (ρ_j) will be grouped into two groups by applying the SymDIRECT algorithm corrected by the ordinary k-means algorithm. The set of lines with greater densities will be denoted by $\hat{\ell}_1, \ldots, \hat{\ell}_r$.

By applying the KCL algorithm to lines $\hat{\ell}_1, \ldots, \hat{\ell}_r$ in Step 7 we obtain a partition $\Pi^\star(r) = \{\pi_1^\star, \ldots, \pi_r^\star\}$ with corresponding line-centers $\ell_1^\star, \ldots, \ell_r^\star$ (see Fig 4c), which will be called the *First Solution*. Note that in this example the First Solution consists of four lines, whereby for one subset of the data set $\mathcal{A}$ a corresponding line has not been recognized (see Fig. 4c). Apart from that, the original data set in that example stems from five lines. This case is a motivation for extending the algorithm in Subsection 3.2.

Example 3.1. The described method of searching for the First Solution will be tested [1] on 100 sets of data generated as in Subsection 2.3. Results, i.e. the number of detected lines and necessary CPU-time, are given in Table 1. The following is specifically stated: CPU-time required for running the SymDIRECT algorithm while searching for border points in Step 2, CPU-time required for one iteration of the KCL algorithm in Step 4-5, CPU-time required for running the SymDIRECT algorithm and the ordinary k-means algorithm in Step 6, and CPU-time required for running the KCL algorithm in Step 7.

[1] All evaluations were done on the basis of our own *Mathematica*-modules freely available at: https://www.mathos.unios.hr/images/homepages/scitowsk/LINES-2.rar, and were performed on the computer with a 2.90 GHz Intel(R) Core(TM)i7-75000 CPU with 16GB of RAM.

No. of	Lines detected							CPU-time				
lines	0	1	2	3	4	5	6	(Step 2)	(Step 4-5)	(Step 6)	(Step 7)	Total
2	0	0	19	0	1	0	0	0.116	0.017	0.065	0.067	0.265
3	0	0	3	20	1	1	0	0.193	0.047	0.074	0.114	0.428
4	0	0	0	4	21	2	0	0.202	0.089	0.096	0.200	0.587
5	0	0	0	0	6	18	4	0.299	0.175	0.102	0.319	0.895

TABLE 1. Characteristics of the First Solution

As can be seen in Table 1, the First Solution itself has achieved good results: a set of original lines was detected in 78% of cases, and total CPU-time is extremely small. However, the set of line-centers of the cluster $\pi_1^\star, \ldots, \pi_{\hat{k}}^\star$ sometimes does not include all lines we search for, and sometimes these lines do not correspond to the original ones.

3.2. *Globally optimal solution.* As an example considered to be an illustration of our algorithm we will assume that we have found a globally optimal partition if line-centers of that partition correspond to (up to some small ϵ) the original lines.

In some cases, due to a large number of lines in the set $\mathcal{P}$ and the distribution of elements in the set $\mathcal{A}$ according to the minimal distance principle, the density of points by individual lines may be distorted. Hence, in addition to the optimal partition $\Pi^\star(r)$, we will consider two more optimal partitions, i.e. the partition $\Pi^\star(r-1)$ with $r-1$ lines and the partition $\Pi^\star(r+1)$ with $r+1$ lines. This will be done such that in a sequence of sorted densities we choose the $r-1$, i.e. $r+1$, largest and partitions associated thereto, to which we apply the KCL algorithm.

After that, it will be necessary to decide which of the three optimal partitions, i.e. $\Pi^\star(r-1)$ (Fig. 5a), $\Pi^\star(r)$ (Fig. 5b), $\Pi^\star(r+1)$ (Fig. 5c), is a partition with the most appropriate number of clusters with line-centers.

3.2.1. *The choice of the most appropriate number of clusters.* It has been shown that, when a cluster-center is not a point but a line segment, none of the known indexes (Davies-Bouldin, Calinski-Harabasz, Dann, the Simplify Silhouette Width Criterion [23]) has yielded acceptable results in relation to our problem. The reason for this lies in the fact that these indexes were constructed for spherical or elliptical clusters. In this paper, we will define a special index for recognizing a partition with the most appropriate number of clusters if cluster-centers are segments with the initial and end point on different sides of the rectangle Δ.

Generally, let $\Pi = \{\pi_1, \ldots, \pi_k\}$ be an optimal partition of the set $\mathcal{A} = \{a^i = (x_i, y_i) \colon i = 1, \ldots, m\} \subset \mathbb{R}^2$ whose cluster-centers are lines ℓ_j, $j =$

$1,\dots,k$. For every cluster π_j, we will define the density

$$\rho_j = \frac{|\pi_j|}{|\ell_j|}, \tag{3.1}$$

where $|\pi_j|$ is the number of elements of the cluster π_j, and $|\ell_j|$ is the length of the visible part of the line in the rectangle Δ.

Note that the cluster, whose center-line lies in the set of lines, from which the data come, will have a relatively greater density. Due to mutual intersection of lines, this construction may have minor deviations. Note also that grouping of pairs (π_j, ℓ_j), $j = 1,\dots,k$ by densities $\rho_1,\dots,\rho_k$ into two groups can lead to recognition of some of k lines.

Furthermore, if we denote a sequence of densities by $\rho = (\rho_1,\dots,\rho_k)$, and a variance of the sequence ρ by

$$\texttt{Var}(\rho) = \tfrac{1}{k-1}\sum_{j=1}^{k}(\rho_j - \bar\rho)^2, \quad \bar\rho = \tfrac{1}{k}\sum_{j=1}^{k}\rho_j, \quad k \ge 2, \tag{3.2}$$

then we can define the Geometrical Objects index (GO) in the case of lines as cluster-centers

$$\texttt{GO}(k) := \texttt{Var}(\rho). \tag{3.3}$$

A lower value of the GO index reveals a better partition.

Example 3.2. For the data given in Example 3.1, optimal partitions $\Pi^\star(r-1)$, $\Pi^\star(r)$, and $\Pi^\star(r+1)$ will be defined as described previously, and based upon the GO index, we will decide which one of them is the closest to the globally optimal one. The partition with the most appropriate number of clusters will be the one for which the GO index assumes the smallest value. In the example that was used to illustrate the method, it is a 5-partition shown in Fig. 5c.

(a) $\texttt{GO}(3) = 31.91$ (b) $\texttt{GO}(4) = 13.29$ (c) $\texttt{GO}(5) = 0.11$

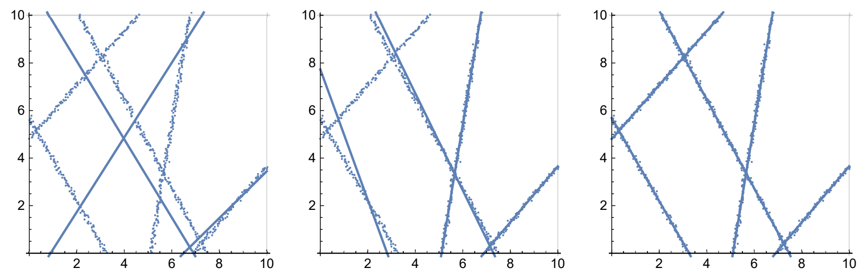

Figure 5. The choice of a partition with the most appropriate number of clusters

Results, i.e. the number of detected lines and necessary `CPU`-time, are given in Table 2. The following is specifically stated: `CPU`-time required for obtaining the First Solution and `CPU`-time required for running the `KCL` algorithm.

No. of	Lines detected							`CPU`-time		
lines	0	1	2	3	4	5	6	(First Solution)	(`KCL`)	Total
2	0	0	19	0	1	0	0	0.265	0.110	0.375
3	0	0	0	24	0	1	0	0.428	0.306	0.734
4	0	0	0	0	27	0	0	0.587	0.514	1.102
5	0	0	0	0	1	26	1	0.895	0.795	1.689

Table 2. Characteristics of a globally optimal partition

As can be seen in Table 2, the percentage of recognition is very high, and total `CPU`-time remains small.

3.2.2. *Analysis of cases with non-detected lines.* Particularly, we will consider examples of data sets where our method did not detect an optimal partition. These examples are shown in Fig. 6.

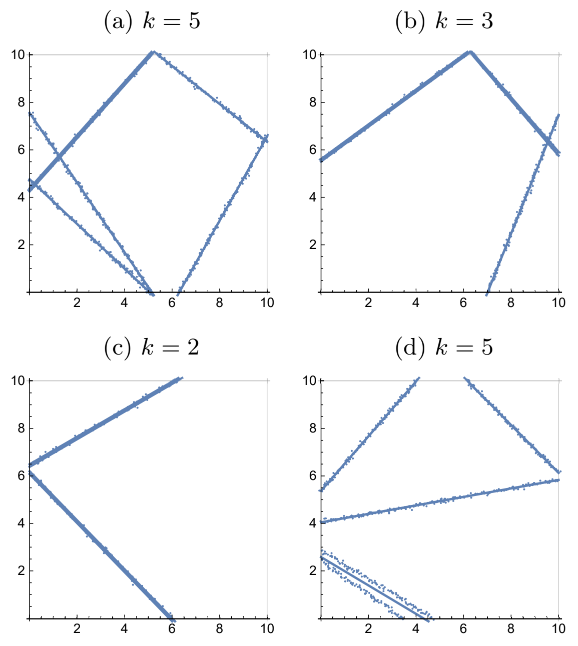

Figure 6. The cases with non-detected lines

The data set $\mathcal{A}$ in Fig. 6a comes from 5 lines, and our method has detected 6 lines. Two lines have appeared whose Hausdorff distance in the rectangle Δ is 0.127, and their slopes differ only in the fourth decimal place.

The data set $\mathcal{A}$ in Fig. 6b comes from 3 lines, and our method has detected 5 lines. Two pairs of lines have appeared whose Hausdorff distances in the rectangle Δ are 0.155 and 0.133 respectively, and their slopes in the first case differ in the second decimal place and in the second case they differ only in the fourth decimal place.

The data set $\mathcal{A}$ in Fig. 6c comes from 2 lines, and our method has detected 4 lines. Two pairs of lines have appeared whose Hausdorff distances in the rectangle Δ are 0.162 and 0.119 respectively, and their slopes in both cases differ in the third decimal place. There is only the example shown in Fig. 6d for which our algorithm does not recognize a globally optimal solution.

It can be seen that a small correction in the algorithm would make these cases well recognized, too. For example, the measure of similarity can be calculated for all pairs of obtained lines by using Hausdorff distance (see [12, 25]). For each pair of similar lines, only one line should be kept. For such a revised set of lines, `KCL` algorithm should be performed once again.

3.3. *Comparison with the incremental method.* The proposed new method will be compared with the `IMLD` algorithm described in Subsection 2.2 on the same 100 data sets as in Example 3.1, i.e. Example 3.2.

No. of	Lines detected							CPU		
lines	0	1	2	3	4	5	6	`DIRECT`	`KCL`	Total
2	0	0	19	0	0	0	1	13.403	0.419	13.822
3	0	0	6	18	1	0	0	14.157	0.949	15.106
4	0	0	11	2	14	0	0	20.844	1.929	22.773
5	0	0	12	5	1	9	1	29.090	3.217	32.308

TABLE 3. The `IMLD` algorithm

Results, i.e. the number of detected lines and necessary `CPU`-time, are given in Table 3. The following is specifically stated: `CPU`-time required for running the `DIRECT` algorithm in `Step 2` and `CPU`-time required for running the `KCL` algorithm in `Step 3`.

As can be seen in Table 3, necessary `CPU`-time is significantly longer and the percentage of recognition is considerably lower, especially for cases with many lines.

4. CONCLUSIONS

The multiple line detection problem occurs in different application areas where it is particularly important to construct an algorithm that can solve such problems in real time. The method we have proposed in this paper solves this problem quickly, precisely and with an extremely high percentage of recognition. With a small software correction, cases shown in Fig. 6a,b,c can also be considered as resolved and we can say that our method had only 1% of non-detected cases. With appropriate software optimization, the proposed method could certainly be utilized in applications where a real-time solution is expected.

For center-based clustering methods used for recognizing geometrical objects, it is important to have an efficient index for recognizing a partition with the most appropriate number of clusters. A new special index has been proposed in this paper for cases where cluster-centers are lines. This index could be easily used in other similar situations.

Acknowledgements.

The author would like to thank the referees and the journal editors for their careful reading of the paper and insightful comments that helped us improve the paper. Especially, the authors would like to thank Mrs. Katarina Moržan for significantly improving the use of English in the paper. This work was supported by the Croatian Science Foundation through research grants `IP-2016-06-6545` and `IP-2016-06-8350`.

References

[1] A. M. Bagirov, J. Ugon and H. Mirzayeva, *Nonsmooth nonconvex optimization approach to clusterwise linear regression problems*, European J. Oper. Res. **229** (2013), 132–142.

[2] A. M. Bagirov, J. Ugon and D. Webb, *Fast modified global k-means algorithm for incremental cluster construction*, Pattern Recognition **44** (2011), 866–876.

[3] M. Barni and R. Gualtieri, *A new possibilistic clustering algorithm for line detection in real world imagery*, Pattern Recognition **32** (1999), 1897–1909.

[4] R. Duda, P. Hart and D. Stork, Pattern Classification, Wiley, 2001.

[5] L. A. Fernandes and M. M. Oliveira, *Real-time line detection through an improved Hough transform voting scheme*, Pattern Recognition **41** (2008), 299–314.

[6] C. Fernández, V. Moreno, B. Curto and J. A. Vicente, *Clustering and line detection in laser range measurements*, Robotics and Autonomous Systems **58** (2010), 720–726.

[7] R. Grbić, E. K. Nyarko and R. Scitovski, *A modification of the DIRECT method for Lipschitz global optimization for a symmetric function*, J. Global Optim. **57** (2013), 1193–1212.

[8] D. R. Jones, C. D. Perttunen and B. E. Stuckman, *Lipschitzian optimization without the Lipschitz constant*, J. Optim. Theory Appl. **79** (1993), 157–181.

[9] J. Kogan, Introduction to Clustering Large and High-dimensional Data, Cambridge University Press, New York, 2007.

[10] V. Leemans and M.-F. Destain, *Line cluster detection using a variant of the Hough transform for culture row localisation*, Image and Vision Computing **24** (2006), 541–550.

[11] A. Manzanera, T. P. Nguyen and X. Xu, *Line and circle detection using dense one-to-one Hough transforms on greyscale images*, EURASIP Journal on Image and Video Processing **2016**, Article number: 46 (2016), DOI `10.1186/s13640-016-0149-y`.

[12] T. Marošević, *The Hausdorff distance between some sets of points*, Math. Commun. **23** (2018), 247–257.

[13] P. Mukhopadhyay and B. B. Chaudhuri, *A survey of Hough transform*, Pattern Recognition **48** (2015), 993–1010.

[14] Y. Nievergelt, *Total least squares: state-of-the-art regression in numerical analysis*, SIAM Rev. **36** (1994), 258–264.

[15] K. Sabo and R. Scitovski, *An approach to cluster separability in a partition*, Inform. Sci. **305** (2015), 208–218.

[16] K. Sabo, R. Scitovski and I. Vazler, *One-dimensional center-based l_1-clustering method*, Optim. Lett. **7** (2013), 5–22.

[17] R. Scitovski, *A new global optimization method for a symmetric Lipschitz continuous function and application to searching for a globally optimal partition of a one-dimensional set*, J. Global Optim. **68** (2017), 713–727.

[18] R. Scitovski and S. Scitovski, *A fast partitioning algorithm and its application to earthquake investigation*, Computers & Geosciences **59** (2013), 124–131.

[19] H. Späth, *Algorithm 48: a fast algorithm for clusterwise linear regression*, Computing **29** (1982), 175–181.

[20] H. Späth, Cluster-Formation und Analyse, R. Oldenbourg Verlag, München, 1983.

[21] J. C. R. Thomas, *A new clustering algorithm based on k-means using a line segment as prototype*, in: CIARP, 2011, Lecture Notes in Comput. Sci. 7042, pp. 638–645.

[22] H.-H. Trinh, D.-N. Kim and K.-H. Jo, *Facet-based multiple building analysis for robot intelligence*, Appl. Math. Comput. **205** (2008), 537–549.

[23] L. Vendramin, R. J. G. B. Campello and E. R. Hruschka, *On the comparison of relative clustering validity criteria*, in: Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 – May 2, 2009, Sparks, Nevada, USA, SIAM, 2009, pp. 733–744.

[24] I. Vidović, D. Bajer and R. Scitovski, *A new fusion algorithm for fuzzy clustering*, Croat. Oper. Res. Rev. CRORR **5** (2014), 149–159.

[25] I. Vidović and R. Scitovski, *Center-based clustering for line detection and application to crop rows detection*, Computers and Electronics in Agriculture **109** (2014), 212–220.

Brza i učinkovita metoda za prepoznavanje više pravaca u ravnini

Rudolf Scitovski, Una Radojičić i Kristian Sabo

Sažetak. U ovom radu, promatramo problem prepoznavanja više pravaca u ravnini na osnovi podataka poteklih od većeg broja pravaca, koji nisu unaprijed poznati. Predložena je nova i učinkovita metoda, utemeljena na grupiranju podataka na bazi centara, koja ovaj problem rješava brzo i precizno. Metoda je testirana na 100 slučajno generiranih skupova podataka. U usporedbi s inkrementalnom metodom, predložena metoda daje značajno bolje rezultate. Također, u cilju određivanja particije s najprihvaljivijim brojem grupa, predložen je novi indeks, za slučaj u kome su pravci središta grupa. Indeks se također može poopćiti na druge geometrijske objekte.

Rudolf Scitovski
Department of Mathematics
University of Osijek
31 000 Osijek, Croatia
E-mail: scitowsk@mathos.hr

Una Radojičić
Department of Mathematics
University of Osijek
31 000 Osijek, Croatia
E-mail: uradojic@mathos.hr

Kristian Sabo
Department of Mathematics
University of Osijek
31 000 Osijek, Croatia
E-mail: ksabo@mathos.hr

Received: 30.10.2018.
Revised: 20.1.2019.
Accepted: 19.2.2019.