Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 3532 – 3543

doi:10.3906/elk-1809-157

An algorithm for line matching in an image by mapping into an n-dimensional vector space

Raiymbek SULTANOV[1], **Ahmet ATAKAN**[2], **Rita ISMAILOVA**[1*]
[1]Department of Computer Engineering, Faculty of Engineering, Kyrgyz-Turkish Manas University, Bishkek, Kyrgyz Republic
[2]Department of Computer Engineering, Graduate School of Natural and Applied Science, Middle East Technical University, Ankara, Turkey

Received: 21.09.2018 • **Accepted/Published Online:** 29.03.2019 • **Final Version:** 18.09.2019

Abstract: This paper proposes a minimal length difference algorithm for construction of a line in an image by solving the problem of optimal contour approximation. In this algorithm, a method for finding interest points is proposed, and the object matching (classification) is done by mapping interest points onto a vector space. In cases where the lines in the representation of the images are not smooth, the algorithm converges rapidly. The results of the experiments showed that for convergence of the contour simplification, there were 5-6 iterations for $n = 13$. To check how close the curve approximation calculated by the algorithm above, the researchers have calculated the length of the curve simplification manually. This length was then compared to the length of the original curve. The results showed that the length of the simplified curve grew rapidly to 92%–95% of the original curve length. The further increase in the number of points does not affect this indicator. According to the obtained results, the relative difference and the relative difference distance are good metrics to match objects.

Key words: Shape simplification, interest points, local features, line simplification, pattern matching

1. Introduction

The problem of object detection and recognition is an important problem in computer vision. It is a well-known fact that the lines of image contours can determine the shape of an object in an image. The shape of an image can further be used for matching objects by comparing it with an object model. In [1], it is stated that there are two approaches for image comparison, which are based on the color and texture of an image or on the geometric shape of an object. To compare the object on an image within the given object model, in the shape-based approach, the identical segments on the analyzed shapes' contours are searched for each segment of the model. There are several approaches for solving this problem [2]. To reduce the complexity of matching, an approximation of the shape can be represented by a set of points. That is, the original shape is replaced by simplification of a curve—the set of points, which are connected by straight lines—a polygon. In this case, the distance between the original polygonal chain and the approximated one is measured. This process is called line simplification. Since shape contours can be considered as lines, line simplification can refer to contour simplification in the context of object recognition. The results of line simplification depend on the characteristics of a shape [3]. As the number of points increases, the approximation result gets closer to the original shape. However, increased point number also increases the complexity of the algorithm. Douglas and

*Correspondence: rita.ismailova@manas.edu.kg

Peucker proposed a method for the contour simplification in 1973, where they build a polygon by calculating integrals of difference between the possible shortcuts and the original contour to find the minimal value among these integrals [4]. Thus, one of the main problems in simplification is finding so-called interest points, from which polygons are constructed. Interest points are unique characteristics of an object that allows comparing the object with itself or with similar object classes. That is, one needs to find a set of points in an object contour and therefore define the object by drawing lines through these set points. In [5], the interest points were defined from contour points where signal changes. Finding interest points of an object contour, which can precisely describe an image, is a crucial phase in object recognition problem of image processing. There are several ways to distinguish such points.

This paper proposes a new model to match objects by mapping polygon lines, obtained by contour simplification, onto n-dimensional space. The idea of the method is based on the fact that in an object, a ratio of line lengths are invariant under compression (changing the shape size while retaining the ratio) and shifting. The algorithm proposed in this paper determines points in a unique way so the distances between these points are also invariant. By defining these invariants as coordinates in space, one should create a mapping of lines onto a vector space. Objects are matched by comparing defined invariants.

2. Related works

One of the algorithms, which is widely used for contour approximation [6], has been proposed by Douglas and Peucker [4]. The idea behind this algorithm is to find the first and last points in a curve, which are joined by a straight (base) line. Other points are calculated by finding the point in the curve, which is at the maximal distance from the line and drawing the line through this point, recursively. However, the original algorithm does not preserve the nonintersection property. In addition, in the worst case, the running time of the algorithm can be up to $O(n^2)$.

Various algorithms based on the calculation of distances have been proposed. Some algorithms were to improve the topological consistency [7–13]. However, due to high time complexity, the original Douglas–Peucker algorithm is not suitable for most applications. Thus, modifications were done to improve the speed of the algorithm. For example, Hershberger and Snoeyink [14] proposed a modified path hull based on the geometric structure, which allowed to decrease the running time of the algorithm to $O(n \log n)$. In [15], $O(n \log n)$-time algorithm was proposed to approximate a piecewise linear curve utilizing binary search technique. However, the authors say that some challenges still remain, such as finding a simple approximate curve. Moreover, in [16], a method to reduce the number of points for line approximation was proposed. One of the latest articles in this area by Rodrigues [17] offers a method for calculating distances based on the combination of the Minkowski and Chebyshev distances. Nowadays, this method is widely used in computer vision applications [18, 19]. Another modification of the Douglas–Peucker algorithm, where the division is done with respect to the highest error instead of recursive splitting was proposed in [20]; however, the speed of an algorithm was slower compared to the original one. Li et al. proposed a new approach that combined the Douglas–Peucker algorithm and Li-Openshaw algorithm for smoothing sharp corners [21]. As it can be seen, in shape simplification algorithms mostly the error rate between the original shape and its simplified versions are calculated. The efficiency of algorithms is rated based on this error rate and the speed of algorithms.

3. Method

Mathematically, the Douglas–Peucker algorithm computes the following integral [4]:

$$\min_{x_0<x_1<\cdots<x_k} \sum_{i=0}^{k-1} \int_{x_i}^{x_{i-1}} [f(x) - L(x)]dx, \tag{1}$$

where $f(x)$ is a given curve and $L(x)$ is its simplification such that $f(x) = L(x)$ for the points x_i, $i = 0, 1, \ldots, k$. Here, $x_i \in \mathbb{R}$. In this work, we propose a new method with a different approach to find the points (vertices) for curve simplification. Consider the problem of line matching by mapping it onto n-dimensional space. It is known that the length of a line is invariant under compression and coordinate transfer. The aim is to construct an algorithm for finding certain points on a line in a unique way so that the distances between these points are also invariant. Defining these invariants as coordinates in space, one creates a mapping of lines onto a vector space. That is, invariants refer to the length of linear segments in curve simplifications, and the distances between the vertices and the geometric center of the object. The coordinates of the vector space allow one to identify and classify the lines in the object. To construct such an algorithm, one should consider the following problem: Given a rectifiable line $f(x)$ and an integer $k > 0$, it is required to find the points x_i with $i = 0, 1, 2, \ldots, k+1$ on the line $f(x)$, which satisfy the maximum conditions:

$$\max_{x_0<x_1<\cdots<x_k} \sum_{i=0}^{k-1} \sqrt{(f(x_{i+1}) - f(x_i))^2 + (x_{i+1} - x_i)^2} \tag{2}$$

and the geometric center (x_c, y_c):

$$x_c = \frac{\sum_{i=0}^{k+1} x_i}{k+1}, \quad y_c = \frac{\sum_{i=0}^{k+1} f(x_i)}{k+1} \tag{3}$$

Next, one needs to calculate the distance between points, which is the linear segments' lengths, by solving the problems (2) and (3) as

$$a_i = \sqrt{(x_{i+1} - x_i)^2 + (f(x_{i+1}) - f(x_i))^2},$$

where $i = 0, 1, 2, \ldots, k$, and

$$a_{k+i+1} = \sqrt{(x_c - x_i)^2 + (f(x_c) - f(x_i))^2},$$

where $i = 0, 1, 2, \ldots, k+1$. For the values of a_i, compute b_i as follows:

$$b_i = \frac{a_i}{S_1}, \tag{4}$$

where $i = 0, 1, 2, \ldots, k$, and

$$b_i = \frac{a_i}{S_2}, \tag{5}$$

where $i = k+1, k+2, \ldots, 2k+1$. Here,

$$S_1 = \sum_{i=0}^{k} a_i$$

and

$$S_2 = \sum_{i=0}^{k} a_{k+i+1}.$$

These values determine invariants of contour simplification. For each rectifiable line $f(x)$ on the plane, a point with coordinates b_i on $n = 2k+1$-dimensional vector space is defined. It is obvious that if the problem (2) has a unique solution, then the solution of the problems (2) and (3) uniquely maps the line onto an n-dimensional vector space. Thus, this mapping can be used to identify the line $f(x)$. In practice, the problem can be solved given the first k coordinates. In this work, without loss of generality, only the lengths of the linear segments in simplification of a curve are considered. The values of lengths are invariants of that curve. The method also assumes that if the problem (2) is solved with less than k points, then the remaining coordinates are assumed to be equal to zero. For example, if $f(x)$ is a polygonal chain, solution for the problem (2) can be found with less than k point. If the line consists of r polygonal chains, then the first r coordinates are determined by the total lengths of the polygonal chains, and the remaining $k-r$ coordinates are equated to zero since the solution of the problem (2) does not improve further for any $k < r$. Obviously, for lines in the object that are specified only in certain pixels, it is not easy to find solutions to the problem (2), since the problem (2) can have more than one local maxima. The gradient methods may not be applicable, as the derivatives with respect to the coordinates are very noisy due to the pixel-based line representation in the square grid. Therefore, as experiments showed, a simple iterative method proposed in this paper converges quickly to a global maximum.

Since the lines are given in pixels, it is convenient to represent the line in parametric form as $X(t)$, $Y(t)$, where $0 \le t \le T$. Here, T is the number of pixels, and $X(t)$, $Y(t)$ are the coordinates of the lines represented by the pixels. For closed curves, it is assumed that $X(0) = X(T)$ and $Y(0) = Y(T)$. Note that for closed curves the number of invariants is even, while for nonclosed curves it is odd. The function $\rho(t)$ is defined as the total distance of the linear segments lying between the points $(X(t_{i-1}), Y(t_{i-1})); (X(t), Y(t))$ and $(X(t_{i+1}), Y(t_{i+1})); (X(t), Y(t))$, that is,

$$\rho(t) = \sqrt{(X(t_{i-1}) - X(t))^2 + (Y(t_{i-1}) - Y(t))^2} + \sqrt{(X(t_{i+1}) - X(t))^2 + (Y(t_{i+1}) - Y(t))^2}.$$

Thus, the algorithm for solving the problem (2) can be formulated as follows:

1. The initial approximation is taken uniformly with respect to the parameter t: $t_i = \text{int}(h \cdot i)$, where $i = 0, 1, 2, \ldots, k$, and $h = \frac{T}{k}$;

2. New values of t_i^*, $i = 1, 2, \ldots, k-1$ are calculated by approximation from the condition for maxima for fixed values of t_{i-1} and t_{i+1}

$$\max_{t_{i-1} < t < t_{i+1}} \rho(t)$$

That is, we calculate $\rho(t)$ for t_{i-1} and t_{i+1} and set t^* to maximum of these three values:

$$t^* = \operatorname*{argmax}_{t_{i-1} < t < t_{i+1}} \rho(t)$$

3. Check if values meet the threshold by calculating the difference between the previous approximation point and the calculated one:

$$\frac{|t_i - t_i^*|}{(t^* - t_{i-1}) + (t_{i+1} - t^*)} = \frac{|t_i - t_i^*|}{t_{i+1} + t_{i-1}} < \epsilon.$$

In the experiments, the threshold was set to 0.03. When considering the value of ϵ in pixels, the difference $t_{i+1} - t_i \approx 100$; thus, the value of ϵ in pixels can be taken in the range (3,4,5) and $|t_i - t_i^*| < \epsilon$.

4. If condition for ϵ is not satisfied, go back to step 2; otherwise, the obtained parameters are solution for the problem (2). If

$$\left| \frac{Y(t^*) - Y(t_{i-1})}{Y(t_{i+1}) - Y(t_{i-1})} - \frac{X(t^*) - X(t_{i-1})}{X(t_{i+1}) - X(t_{i-1})} \right| < \epsilon$$

then the point should be removed since it lies in one straight line between t_{i-1} and t_{i+1}. This condition prevents the appearing of duplicated values of t_i as well.

Note that $t_0^* = t_0$ and $t_k^* = t_k$ for $i = 0$ and $i = k$, since values of the starting and ending points are fixed. If an object that one is trying to classify is a closed curve, then the problem (2) should be solved for each pixel starting with the point $X(0), Y(0)$, and the solution with maximal length should be chosen. For example, for a circle there may be as many points as pixels, by which a circle is represented. However, the representation of a circle on the n-dimensional space is unique, and it coincides with an n-uniform polygon. Although the lines in the representation of the objects is not smooth, the algorithm converges rapidly. If the number of points is set to (n), then the proposed algorithm calculates the best polygon approximating the object contour. The results of the experiments showed that there are 6 iterations needed for convergence of the contour simplification for $n = 13$. To check the quality of the curve simplification approximation using the above algorithm, we calculated the length of the simplification of a curve. This length was then compared to the length of the original curve. The results showed that the length of the simplification of a curve grew rapidly to 92%–95% of the original curve length. The further increase in the number of points (in fact, the dimensionality of the vector space) does not affect this indicator. It can be explained by the fact that a circle and an n-dimensional uniform polygon represent the same point in an n-dimensional vector space; they differ in approximating the length of a curve simplification. That is, for an n-dimensional uniform simplification of a curve it is equal to 100%, while for a circle the approximation rate is smaller. Obviously, for the lines in the object, the amount of the points t is the best vector space of minimal dimension, which cannot be improved, and which is what the above algorithm produces. We propose to name this algorithm as the minimal length difference algorithm.

4. Results

4.1. Matching of objects under rotation

In this study, several objects have been selected and identified using the proposed algorithm. First, the special points (vertices) were found for simplification of object contours. Next, lengths of simplification obtained by approximation were calculated. The value for k was taken as 13 for the set of experiments conducted on selected objects. The first object was a polygonal chain, as presented in Figure 1a. Since the special points in a polygonal chain are the endpoints of each straight line, the algorithm defined 5 special points. The 6th point is the geometric center of the object. Next, the image was rotated for 90°, 180°, and 270°. At each rotation, algorithm redefines the special points on the polygonal chain and the geometric center of the object and calculates invariants. If all the distances are equal to that of the previous object, the objects are identified as the same image.

As it was mentioned before, invariants are defined in two ways: as distances between endpoints, and as distances between points and the geometric center of the polygonal chain. In Table 1, distances between endpoints are given. In Table 2, there are invariants, defines as distances between points and the geometric

Table 1. Invariants, defined by (4) of object (Figure 1a) when rotated to 90°, 180°, and 270°.

Angle of rotation	Invariants			
0°	0.257914	0.206538	0.252611	0.282937
90°	0.257787	0.205352	0.253001	0.28386
180°	0.255853	0.206129	0.254581	0.283438
270°	0.255681	0.205283	0.255011	0.284025

center of the polygonal chain are given. As is can be seen from Tables 1 and 2, invariants have a small error rate; thus, the algorithm defines objects as rotated versions of one object.

Table 2. Invariants, defined by (5) (Figure 1a) when rotated to 90°, 180°, and 270°.

Angle of rotation	Invariants				
0°	0.247489	0.160513	0.150891	0.226741	0.214366
90°	0.24897	0.159811	0.149955	0.22723	0.214034
180°	0.248741	0.157833	0.153645	0.225864	0.213918
270°	0.24895	0.156153	0.153409	0.226905	0.214583

Thus, the algorithm uniquely determines the contour simplification edges and identify an object by calculating the distances between vertices and distances between vertices and the geometric center of an object.

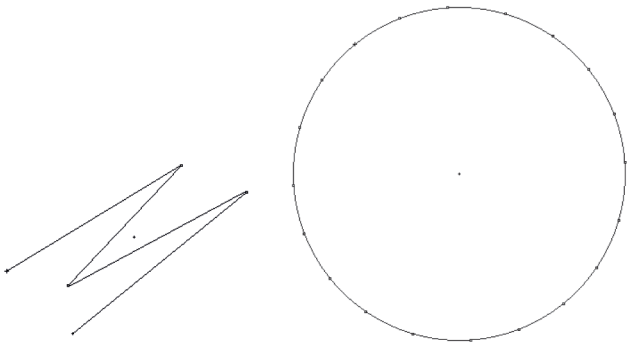

Figure 1. Nonclosed shapes and convex shapes. (a) M-shape, (b) Circle.

For simple objects, such as a circle (Figure 1b), the result of the algorithm is trivial. As it was stated before, a curve can be defined uniquely as the number of special points, which are used to approximate the curve, increases. Thus, the number of linear segments in simplification of an object also increases. However, after some threshold is achieved, the further approximation have no effect on the quality of matching. In this paper, the quality of matching is defined as a ratio of the original contour length and length of the polygonal chain obtained during the approximation process. Thus, one can look at the quality dynamics as the number of special points. For a circle, the matching quality of 90% was achieved after 7 iterations.

Thus, one can look at the quality dynamics as the number of special points. For a circle, the matching quality of 90% was achieved after 7 iterations (Figure 2). As it is seen, some threshold is reached and the quality approaches 95% and stabilizes. The algorithm defined the number of special points, by which a circle (Figure 1b) can be uniquely recognized, to be equal to 20. Of course, this number depends on the size of an object.

4.2. Finding the special points in concave shapes

In case of concave shapes (that is shapes where there are vertices located inside the outer contour of the shape), finding special points is not that trivial, because an algorithm can miss some edge points in a shape contour. However, the proposed algorithm showed a good performance in determining special points in concave shapes as well. For a heart shape (Figure 3), 23 special points and the geometric center of the shape were defined. To evaluate the effectiveness of the proposed algorithm, the shape was rotated to 90° and compressed two times. Next, invariants were calculated for the modified version of the shape.

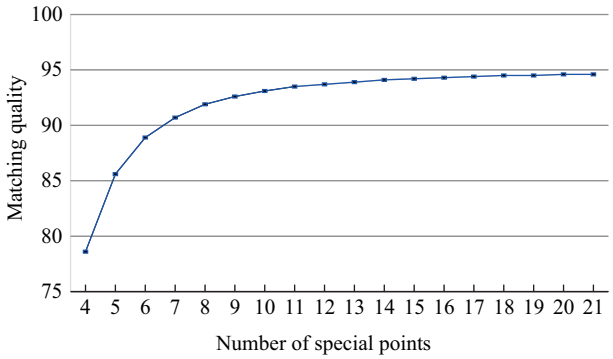

Figure 2. The matching quality of a circle (Figure 1b) when the number of special points increases.

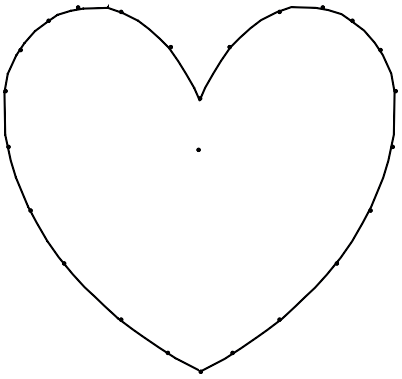

Figure 3. A heart object.

The result of matching the heart shape with its modified version is presented in Figure 4. It can be seen from the figure that calculated invariants show the match of the heart shape and its modified version.

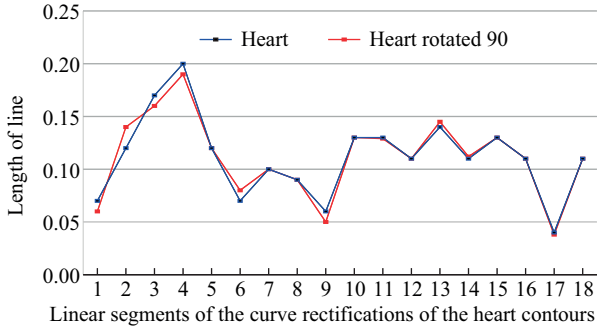

Figure 4. Comparison of invariants of heart object (Figure 3) and its rotated version.

For complex concave shapes, such as the one given in Figures 5a–5c, an algorithm for contour approximation has been run by setting the number of vertices in simplification of a curve to 9 (Figure 5a), to 10 (Figure 5b), and to 31 (Figure 5c). The matching quality above 90% was achieved with 31 special points (Figure 6).

As it can be seen from Figure 6, when the number of special points was set to 10, the monotonicity of the approximation convergence is violated, that is, the length of the curve simplification, given in Figure 5b (with 10 special points) is less than those with 9 special points.

This decrease was due to the fact that in the contour of the object, there are multiple local extrema and solution of the problem depends on the number of special points.

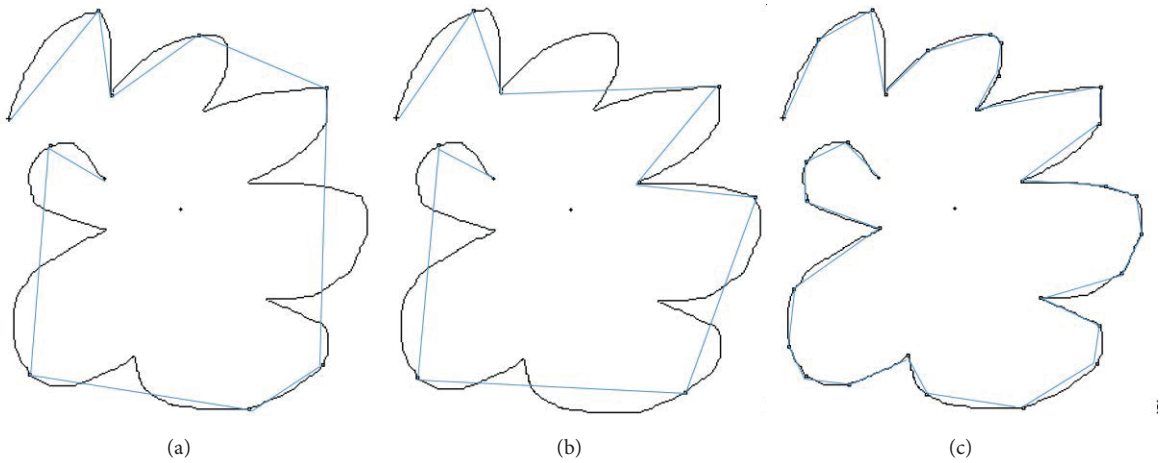

Figure 5. Concave shapes. (a) 9 vertices, (b) 10 vertices, (c) 31 vertices.

4.3. Matching stretched objects

Next, the algorithm was used to compare the object (Figure 7a) and its stretched version (Figure 7b). The star was chosen since in this shape the distances between adjacent edges are equal. These distances are shown in Figure 8 as a straight line (star 1 line). In addition, distances between vertices and the geometric center take only two values (that is, distance between the center and 5 outer vertices, and the center and 5 inner vertices).

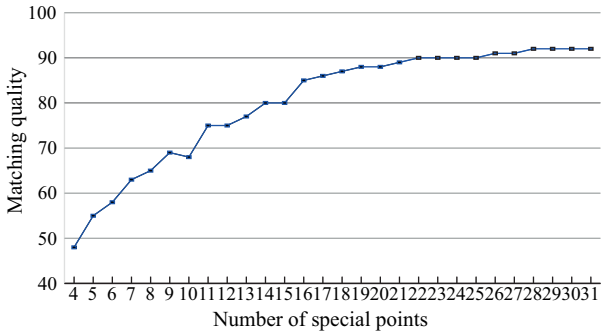

Figure 6. The matching quality of the shape in Figure 5b when the number of special points increases.

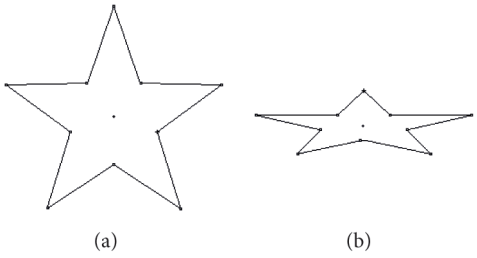

Figure 7. Star: (a) Normal view, (b) Strached view.

However, when the length of adjacent edges in the stretched object are evaluated they are found not to be equal (star 2 line). The same situation was observed while comparing distances between vertices and the geometric center of the object.

The result of comparison of invariants is given in Figure 8. Since the error rates are big enough, the objects were not matched. Thus, the algorithm defines them as two different objects.

4.4. Matching objects with small differences

Next, the algorithm has been run on objects with small differences. For that case, an ellipsoid (Figure 9a) and a callout figure (Figure 9b) were chosen. The number of special points was set to 12.

First, the simplification was done for both for the ellipsoid in Figure 9a and the callout shape in Figure 9b. Next, the algorithm was applied to the callout (Figure 9b) when rotated to 90°, 180°, and 270°. In

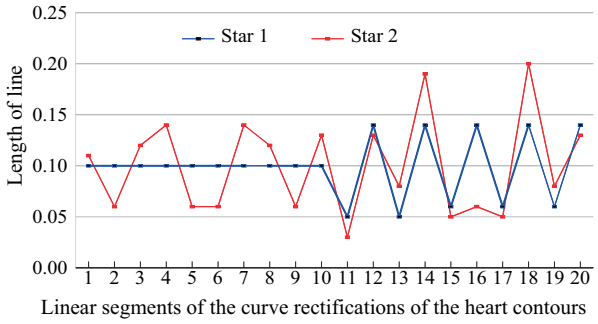

Figure 8. Comparison of invariants of star object (Figure 7a) and its stretched version (Figure 7b).

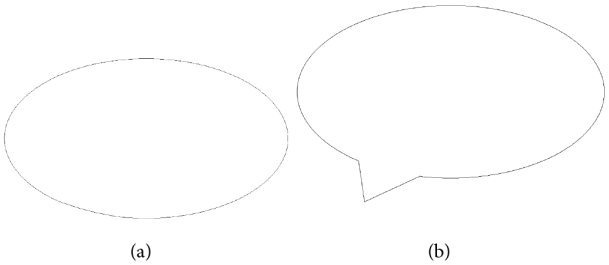

Figure 9. Objects with small difference: (a) Ellipsoid (b) Callout figure

addition, the size of callout shape was reduced two times, and invariant were calculated.

The visualization of resulting invariants of the callout shape and its modified versions, defined by distances between vertices and the geometric centers, are given in Figure 10. Again, as it can be seen from the figure, the standard error rate is quite low (the maximal standard error is 0.004). Thus, the objects were matched.

Next, the average values of invariants were calculated (blue line in Figure 11). The average values were further used to match the callout shape (Figure 9b) and the ellipsoid (Figure 9a). The maximal difference in invariants was equal to 0.0049. The relative difference, calculated as $||a|| = \max_{i=1,\ldots,n} |a_i|$, was equal to 10. The relative difference distance in the norm given by $||a|| = \sqrt{\sum a_i^2}$ was equal to 7. Visually, it was represented in Figure 11.

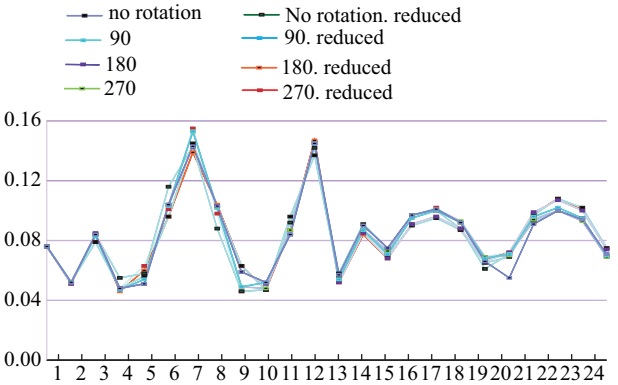

Figure 10. Comparison of invariants for callout shape under rotation and size reduction.

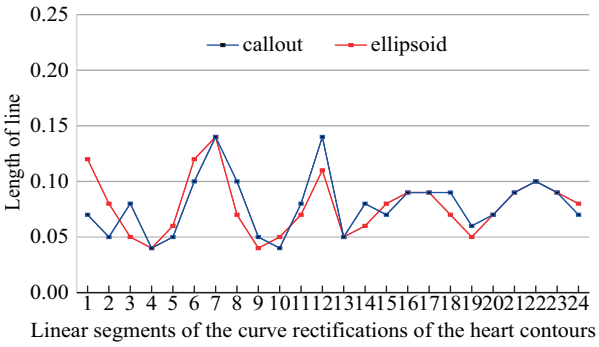

Figure 11. Comparison of invariants of callout object (Figure 9a) and ellipsoid (Figure 9b)

Thus, as the relative difference and the relative difference distance suggests, these figures are not matched.

5. Discussions

For the concave shapes, it is not always easy to find vertices for line simplification with the Douglas–Peucker algorithm since the approximation is done by calculating integrals and finding the minimal value among these integrals. That is, by solving the problem of the form:

$$\min_{x_0<x_1<\cdots<x_n} \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} [f(x) - L(x)]dx.$$

In this paper, an algorithm, where the length of simplification of a curve is approximated and maximized by the defined object, is proposed. That is, in our algorithm, a solution to the problem is proposed as finding:

$$\min_{x_0<x_1<\cdots<x_n} \text{length}(f(x)) - \text{length}(L(x)).$$

Let h denote the height of the triangle drawn when the new point is added in line simplification between points x_{i-1} and x_{i+1}. When the values of n are sufficiently large for the convex smooth curve with no intersection, the solution of the two problems above are identical. It can be seen easily, given the solutions for x_{i-1} and x_{i+1} as it is shown in the Figure 12.

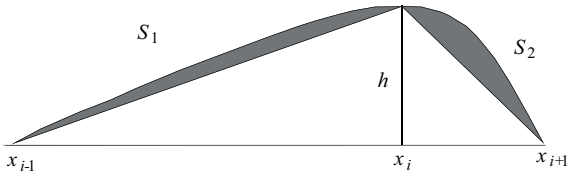

Figure 12. Schematic view when the Douglas–Peucker algorithm and minimal length difference algorithm give identical solutions.

Then, the local problem for the Douglas–Peucker algorithm can be replaced by the problem:

$$\arg_x \min_{h(x)}(S_1(h) + S_2(h)) = \arg_x \min_{x_{i-1}<x<x_{i+1}}(S_1(x) + S_2(x)) = \arg_x \min_{x_{i-1}<x<x_{i+1}} \int_{x_{i-1}}^{x_{i+1}} (f(x) - L(x))dx.$$

On the other hand, for the problem (2), if the local conditions are identical, the length of the simplification is maximal (that is, the difference is the minimal):

$$\arg_x \min_{x_{i-1}<x<x_{i+1}}(\text{length}(f(x)) - \text{length}(L(x))) = \arg_x \min_{h(x)}(S_1(h) + S_2(h)).$$

Thus, the minimal of the integral is achieved when the value of h is maximal. When approximating contours of concave shapes, the efficiency of the proposed algorithm is obvious.

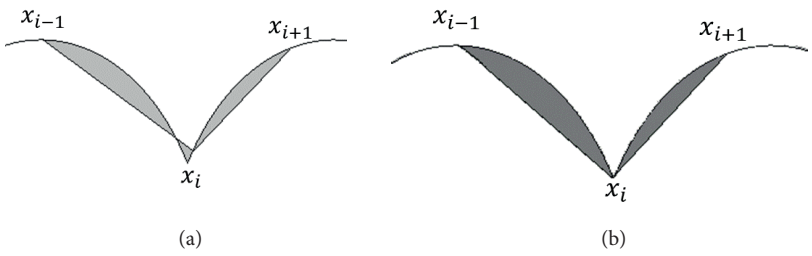

Figure 13. Schematic view of integrals, calculated using: (a) the Douglas–Peucker algorithm and (b) Minimal length difference algorithm.

For example, for a curve in Figure 12, the minimal integral would be achieved if a vertex lied on curve higher than the real breaking point (Figure **??**a), while the correct approximation would be given as it is in Figure **??**b. Since in the proposed algorithm the length of simplification of an object contours is maximized, a higher accuracy is achieved. The complexity of the original Douglas–Peucker algorithm is $O(nm)$, where n is input vertices and m is a number of line segments. Due to the improvement by Hershberger and Snoeyink

[14], the complexity was improved to $O(n \log n)$. Since in the proposed algorithm the same reasoning can be applied, the complexity of our algorithm can also be evaluated as $O(n \log n)$. However, when applied to the closed curves, the number of possible values of the starting point is equal to the number of vertices. That is, approximations are calculated with taking each vertex as a starting point, and the algorithm outputs the solution with the starting point with which the best simplification is achieved. Since the solution with maximal length can be obtained at any of the vertices on the shape, complexity of the algorithm for closed curves becomes $O(n^2)$.

6. Conclusion

According to [22], for the edge detection performance, the low error rate is an essential criterion. In addition, there are other criteria, such as well localization of points at the edges and ease of implementation. As the results showed, invariants can be used as point coordinates in a multidimensional space. Then any simplification of a curve can be uniquely represented as a point in a multidimensional vector space and solve the object matching problem. That is, any simplification of a curve is mapped onto one point and can be used to classify an object. The complexity of the algorithm for nonclosed curves is $O(n \log n)$. However, for closed curves the complexity is $O(n^2)$ since every point at the curve can be taken as a starting point. Thus, the proposed algorithm can be used to solve pattern matching and classification problems. In addition, the algorithm is useful for applications in cartography, since it is efficient for simplification of object contours with the minimal number of special points.

References

[1] Veltkamp RC, Hagedoor M. State of the art in shape matching. In: Lew MS, editor. Principles of Visual Information Retrieval. London, UK: Springer-Verlag, 2001. pp. 87-119.

[2] Alt H, Guibas LJ. Discrete geometric shapes: matching, interpolation, and approximation. In: Sack JR, Urrutia J, editors. Handbook of Computational Geometry. Amsterdam, the Netherlands: Elsevier, 2000. pp. 121-153.

[3] Balboa JLG, López FJA. Sinuosity pattern recognition of road features for segmentation purposes in cartographic generalization. Pattern Recogn 2009; 42: 2150-2159.

[4] Douglas DH, Peucker TK. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartogr: Int J Geogr Inf Geovisualization 1973; 10: 112-122.

[5] Schmid C, Mohr R, Bauckhage C. Evaluation of interest point detectors. Int J Comput Vision 2000; 37: 151-172.

[6] Heckbert PS, Garland M. Survey of polygonal surface simplification algorithms. Tech report, Carnegie-Mellon University, Pittsburgh, PA, USA, 1997.

[7] Saalfeld A. Topologically consistent line simplification with the Douglas–Peucker algorithm. Cartogr Geogr Inf Sc 1999; 26: 7-18.

[8] Wu ST, Marquez MRG. A non-self-intersection Douglas–Peucker algorithm. In: IEEE 2003 16th Brazilian Symposium on Computer Graphics and Image Processing. Sao Carlos, Brazil. Los Alamitos, CA, USA: IEEE; 2003. pp. 60-66.

[9] Park W, Yu K. Hybrid line simplification for cartographic generalization. Pattern Recogn Lett 2011; 32: 1267-1273.

[10] Song X, Cheng C, Zhou C, Zhu D. Gestalt-based Douglas–Peucker algorithm to keep shape similarity and area consistency of polygons. Sens Lett 2013; 11: 1015-1021.

[11] Pallero J L G. Robust line simplification on the plane. Comput Geosci 2013; 61: 152-159.

[12] Du S. Analyzing topological changes for structural shape simplification. J Visual Lang Comput 2014; 25: 316-332.

[13] Tienaah T, Stefanakis E, Coleman D. Contextual Douglas–Peucker simplification. Geomatica 2015; 69: 327-338.

[14] Hershberger JE, Snoeyink J. Speeding up the Douglas–Peucker line-simplification algorithm. In: 5th Symposium on Data Handling. Charleston, SC, USA: IGU Commission on GIS; 1992. pp. 134–143.

[15] Imai H, Iri M. Computational-geometric methods for polygonal approximations of a curve. Comput Vis Graph Image Process 1986; 36: 31-41.

[16] Douglas DH, Peucker TK. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. In: Dodge M, editor. Classics in Cartography: Reflections on Influential Articles from Cartographica. Oxford, UK: Wiley-Blackwell, 2011. pp. 15-28.

[17] Rodrigues ÉO. Combining Minkowski and Cheyshev: New distance proposal and survey of distance metrics using k-nearest neighbours classifier. Pattern Recogn Lett 2018; 110: 66-71.

[18] Perez JC, Vidal E. Optimum polygonal approximation of digitized curves. Pattern Recogn Lett 1994; 15: 743-750.

[19] Bradski G, Kaehler A. Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol, CA, USA: O'Reilly Media, 2008.

[20] Ballard D H, Brown C M. Computer vision. Englewood Cliffs, NJ, USA: Prentice-Hall, 1982.

[21] Li C, Wu P, Gu T, Liu X. A study on curve simplification method combining Douglas–Peucker with Li-Openshaw. In: International Conference on Geo-Informatics in Resource Management and Sustainable Ecosystems. Hong Kong, China. Singapore: Springer; 2016. pp. 286-294.

[22] Canny J. A computational approach to edge detection. IEEE T Pattern Anal 1986; Pami-8: 679-98.