PAPER

Sampling Shape Contours Using Optimization over a Geometric Graph

Kazuya OSE[†*], *Nonmember*, **Kazunori IWATA**[†a)], *and* **Nobuo SUEMATSU**[†**], *Members*

SUMMARY Consider selecting points on a contour in the *x-y* plane. In shape analysis, this is frequently referred to as contour sampling. It is important to select the points such that they effectively represent the shape of the contour. Generally, the stroke order and number of strokes are informative for that purpose. Several effective methods exist for sampling contours drawn with a certain stroke order and number of strokes, such as the English alphabet or Arabic figures. However, many contours entail an uncertain stroke order and number of strokes, such as pictures of symbols, and little research has focused on methods for sampling such contours. This is because selecting the points in this case typically requires a large computational cost to check all the possible choices. In this paper, we present a sampling method that is useful regardless of whether the contours are drawn with a certain stroke order and number of strokes or not. Our sampling method thereby expands the application possibilities of contour processing. We formulate contour sampling as a discrete optimization problem that can be solved using a type of direct search. Based on a geometric graph whose vertices are the points and whose edges form rectangles, we construct an effective objective function for the problem. Using different shape datasets, we demonstrate that our sampling method is effective with respect to shape representation and retrieval.

key words: *contour sampling, shape representation, shape retrieval, geometric graph*

1. Introduction

The contour of a shape is vital information for shape recognition. Examples of contours include line drawings made using a digital pen and boundaries obtained through image segmentation. Instead of using the entire contour to determine the shape, we sometimes select a set of points on the contour to simplify the contour and reduce the computational cost. The selected points are called sample points. Accordingly, it is important to select sample points such that they adequately express the shape of the contour. Using sample points in shape representation has some technical merits. For example, we can take advantage of some useful local descriptors, such as the distance set [1], the mixture of von Mises distributions (vMDs) [2], and the self-closed partial descriptor [3] for shape recognition. Such local descriptors generally have their respective invariants for some geometrical transformations; in addition, they can be used easily because, unlike shape recognition using artificial neural networks, they do not involve a learning process that requires a great deal of training data. Additionally, by constructing a configuration matrix of the sample points, we can perform statistical shape analysis [4], [5], which has been established in mathematical statistics.

Contours are classified into two types: those with a certain stroke order and number of strokes, and those with an uncertain order and number of strokes. Examples of the former are letters, such as the English alphabet and Arabic figures, and examples of the latter are pictures of symbols. Regarding the former, the stroke order and number of strokes are generally informative, and some methods based on this type of contour can be used for sampling the contours. For example, vertex detection using polygonal line approximation (PLA) [5]–[10] can be used to select detected vertices as sample points. Analogously, although dominant point (DP) detection [11]–[14] and high curvature point extraction [3], [15], [16] use different measures to calculate curvature, these methods are essentially the same because they can be used to determine curvature extremum points as sample points.

However, for many applications, the stroke order and number of strokes are unavailable or inconsistent. Hence, vertex detection, DP detection, and high curvature point extraction are restricted in their application because of their dependence on the stroke order and number of strokes. To date, little research has focused on sampling contours that lack a stroke order and number of strokes. The aim of this paper is to provide a contour sampling method that does not depend on the stroke order and number of strokes. Specifically, we represent the contour shape using a geometric graph [17] that consists of a set of vertices and a set of edges. In our geometric graph, a vertex is indicated by a coordinate and an edge is indicated by a rectangle, which differs from the conventions of graph theory [18]. Representing the contour shape using another geometric graph was demonstrated in [19]. However, the sampling objective in [19] was to extract the closed boundary between an entire object and the background, whereas our objective is to obtain a skeletal configuration of sample points for shape representation. Using the geometric graph to construct an objective function, we formulate contour sampling as a discrete optimization problem that can be solved using a type of direct search [20]. Using various shape datasets, we demonstrate that our method selects a suitable configuration of sample points for effective shape representation and retrieval.

This paper is organized as follows: We formulate con-

Manuscript received October 18, 2018.
Manuscript revised July 12, 2019.
Manuscript publicized September 11, 2019.
[†]The authors are with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
[*]Presently, with Inbility Company Limited, Japan.
[**]The author retired from the university on December 4, 2018.
a) E-mail: kiwata@hiroshima-cu.ac.jp
DOI: 10.1587/transinf.2018EDP7353

tour sampling as an optimization problem and then describe an algorithm to solve the problem in Sect. 2. Using several shape datasets, we evaluate our sampling method in Sect. 3. Finally, we summarize our findings in Sect. 4.

2. Problem Formulation

We consider a contour simply as a set of points because the contours that we aim to process do not necessarily have a corresponding stroke order and number of strokes.

2.1 Notation

Let $\mathcal{S}$ be a set of points that correspond to the contours of a shape. We assume that $\mathcal{S}$ is a finite set. Contours whose points consist of pixels in a digital image satisfy this assumption because the total number of pixels in a digital image is finite. When contours correspond to curves in the x-y plane, extracting a finite number of points from the curves yields a finite set. An element in $\mathcal{S}$ is called a contour point. We denote n sample points by $(x_1, y_1), \ldots, (x_n, y_n)$, and assume that $3 \leq n \leq |\mathcal{S}|$.

2.2 Problem

For a given n, the problem of contour sampling corresponds to the optimization problem of selecting n sample points from $\mathcal{S}$. When the number of elements in $\mathcal{S}$ is N, the total number of choices is $\binom{N}{n}$, where $n \leq N$. Clearly, exploiting n sample points instead of $\mathcal{S}$ simplifies and reduces the cost of processing the contour. Depending on N and n, the number of choices can be very large. As a result, we cannot assess all the choices. Accordingly, we present an easy-to-use search method to efficiently determine a suitable selection of points to sample the contours. The search method is based on a combinational approach without derivative information, which is suitable for optimization problems with non-smooth objective functions.

Before discussing our sampling method, we describe where sample points should be located on contours with an uncertain stroke order and number of strokes. As an example, we focus on extracting characteristic pixels from thinned contours processed by a thinning operation [21] for digital image curves. In this case, all the pixels of the thinned contours are classified into end pixels, branch pixels, or other (passing) pixels. Then, the thinned contours are divided into segments based on the end pixels and branch pixels. This implies that the end pixels and branch pixels are characteristic points of the thinned contours. Our sampling method further develops this approach by representing the contours using a graph. Specifically, we regard sample points as vertices of a graph over the contours. Then, we select a vertex on the contours such that it has a high degree (i.e., a high number of edges incident to it [18]) and long edges, thereby effectively expressing the shape of the contours.

2.3 Representation

A geometric graph [17] is a graph in which the vertices or edges are connected by geometric objects. To express the contours, we use a geometric graph whose vertices are sample points in the x-y plane, and whose edges form rectangles with a common width denoted by w. The length of a rectangular edge is determined by the Euclidean distance between the corresponding sample points. Throughout this paper, we refer to the geometric graph as a rectangular edge graph, and denote it by $G(V_n, E)$ when the number of vertices is n. In accordance with the custom of graph theory [18], V_n and E designate a set of n vertices and a set of edges, respectively. Figure 1 shows an example of $G(V_4, E)$. The plus signs represent sample points, and the black and white dots represent contour points in $\mathcal{S}$. We explain the difference between the black and white dots later in this paper. The shaded portions depict rectangular edges with a common width.

The existence of a rectangular edge between a pair of sample points depends on the density of the contour points along the rectangular edge. The density of contour points is measured as follows: We draw a straight line segment between sample points (x_i, y_i) and (x_j, y_j), and arrange a rectangle symmetrically with respect to the line segment. The rectangle width is fixed by a constant w, and the length is determined by the length of the line segment. Figure 2 illustrates the resulting rectangle. Let $D\left((x_i, y_i), (x_j, y_j)\,;\, w\right)$ be the longest interval between adjacent points at which perpendicular lines from all the contour points inside the rectangle intersect the line segment. Then, for any non-negative constant a, if

$$D\left((x_i, y_i),\, (x_j, y_j)\,;\, w\right) < a \max_{(x,y)\in\mathcal{S}} \sqrt{(x-\bar{x})^2 + (y-\bar{y})^2}, \quad (1)$$

then the contour points are dense between (x_i, y_i) and

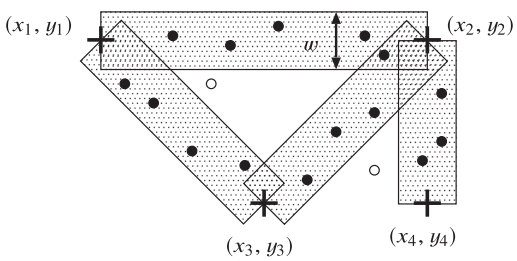


Fig. 1 Example of a rectangular edge graph.

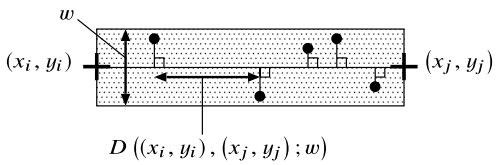


Fig. 2 Rectangle edge between sample points.

(x_j, y_j), and an edge exists between those sample points. Here, $(\bar{x}, \bar{y})$ denotes the center of gravity of the contour points in $\mathcal{S}$; hence, the right-hand side of (1) equates to a times the length between the center of gravity and the furthest point from it. Henceforth, a is referred to as the space-controlling factor. When there are contour points inside a rectangle edge, we say that the edge "covers" those points.

The concept of a rectangular edge is motivated by a criterion of PLA. A polygonal line drawn using the reduced points should provide a good approximation of that drawn using the original points. The rectangular criterion (e.g., see [22]) is used to evaluate the accuracy of the approximation, and supposes that the original points would exist in a rectangular area like the rectangular edge. A similar concept can also be seen in [23]. Using a similar criterion, it attempts to group the rough strokes drawn by a pen, which correspond to the contour points.

2.4 Evaluation Function

For any rectangular edge graph $G(V_n, E)$, we define

$$F((x_1, y_1), \ldots, (x_n, y_n); w) = \sum_{i=1}^{n} f(x_i, y_i; w) \tag{2}$$

to evaluate the location of n vertices, $(x_1, y_1), \ldots, (x_n, y_n) \in V_n$, according to how well their rectangular edge graph represents the contour shape. Function f of (x_i, y_i) given edge width w denotes the degree of contribution of the vertex, and is given by

$$f(x_i, y_i; w) = \left| \mathcal{E}(E; w) \setminus \mathcal{E}\left(E^{(i)}; w\right) \right|, \tag{3}$$

where $|\cdot|$ denotes the number of elements in a set and $\setminus$ denotes the set difference. $\mathcal{E}(E; w)$ is the set of contour points covered by all the edges of the rectangular edge graph, and $\mathcal{E}\left(E^{(i)}; w\right)$ is the set of contour points covered by all the edges of the rectangular edge graph obtained by deleting the i-th vertex from the n vertices. Hence, $f(x_i, y_i; w)$ corresponds to the number of contour points in $\mathcal{S}$ that are covered only by the edges incident to (x_i, y_i). Note that the degree of contribution tends to be large when the i-th vertex has a high degree and long edges, because generally, many and/or long rectangular edges cover many contour points. It is also important to note that evaluation function F is written as a sum of contributions with respect to n vertices.

We provide an example of the degree of contribution using $G(V_4, E)$, referring to Fig. 1. Again, the plus signs represent sample points, and there exist four rectangular edges between (x_1, y_1) and (x_2, y_2), (x_1, y_1) and (x_3, y_3), (x_2, y_2) and (x_3, y_3), and (x_2, y_2) and (x_4, y_4), according to the contour point density. The contour points represented by black dots are covered by the rectangular edges, whereas the contour points represented by white dots are not covered. Then, counting the black dots, we have $|\mathcal{E}(E; w)| = 15$, $\left|\mathcal{E}\left(E^{(1)}; w\right)\right| = 7$; therefore, the degree of contribution of (x_1, y_1) is $f(x_1, y_1; w) = 8$. Analogously, the contributions of (x_2, y_2), (x_3, y_3), and (x_4, y_4) are $f(x_2, y_2; w) = 11$, $f(x_3, y_3; w) = 7$, and $f(x_4, y_4; w) = 3$, respectively.

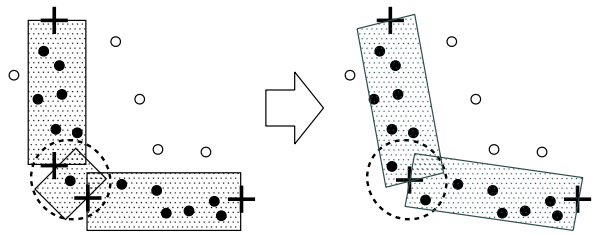


Fig. 3 Replacing pairs of vertices that are too close together with the contour point nearest to their center of gravity.

2.5 Optimization Using Direct Search

In this subsection, we describe our direct search method to optimize the n vertices of a rectangular edge graph. Specifically, for a set of contour points $\mathcal{S}$,

1. Set the number of vertices, the rectangular edge width, and the space-controlling factor, denoted by n, w, and a, respectively.
2. Select n different initial vertices, $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{S}$.
3. Update the n vertices in ascending order of the corresponding degree of contribution. Updating vertex (x_i, y_i) means replacing it with
$$(x', y') = \underset{\substack{(x,y) \in \mathcal{S} \setminus \{(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), \\ (x_{i+1}, y_{i+1}), \ldots, (x_n, y_n)\}}}{\operatorname{argmax}} f(x, y; w). \tag{4}$$
4. Repeat Step 3 while evaluation function F increases.
5. Replace each set of vertices that are too close to each other with the contour point nearest to the center of gravity of the set (see Fig. 3 for an example).

Thus, the evaluation function in (2) corresponds to a non-smooth objective function for the optimization of the n vertices of a rectangular edge graph. Note that the resultant number of vertices is not always n because of the last step. In the experiments below, when the distance between two vertices is less than w, we assess them to be too close. Clearly, this direct search method depends on the initial vertices in Step 2. To reduce this dependency, we generate the initial vertices in Step 2 several times and choose the best initial vertices according to evaluation function F. We provide an illustrative example of updates in Step 3 in Fig. 16.

3. Experiments

We use three shape datasets to compare our sampling method with the uniform sampling method, DP detection [11], and vertex detection using PLA [6], [7]. The uniform sampling method is implemented by taking 10^6 random samples of size n from the set of contour points without duplication and determining the best sample among the

10^6 random samples. The best sample is selected such that the distance between its closest two sample points is maximized. This method serves as the baseline that we use to measure the improvement in contour sampling achieved by our method.

3.1 Time-Series Data of Handwritten Symbol Drawings

(1) Dataset

We created a time-series dataset consisting of 10 handwritten symbol drawings in each of five classes, for a total of 50 symbols, as shown in Fig. 4. This dataset is available from [24]. Each contour in each class was drawn using a different stroke order, number of strokes, and shape scale. Thus, this time-series dataset allows us to examine how the differences in the stroke order and number of strokes affects each sampling method. We set the number of contour points and the space-controlling factor to $N = 100$ and $a = 0.15$, respectively, for each symbol drawing. We set the edge width to $w = 12$ pixels.

(2) Evaluation

We set the number of sample points to $n = 16$ for each symbol drawing. Because we could not specify the number of sample points regarding DP detection, we set the bending-value threshold, which is one of its parameters, to $\varepsilon = 1$ so that the resultant number of sample (dominant) points was around 16. We evaluated the configuration of sample points selected using each sampling method using a visual check and the retrieval rate, which correspond to subjective and objective evaluations, respectively. We computed the retrieval rate using the bullseye test which has been used for shape retrieval [1]–[3]. In the test, each symbol drawing's sample points were used as a query and matched against all the symbol drawings to obtain a matching cost. We then counted the number of correct matches for the top 10 matching symbols for each symbol drawing. Each of the five classes contained 10 symbol drawings; thus, the total number of correct matches was at most 10×50. Thus, we obtained the overall retrieval rate for the top 10 matches by dividing the number of correct matches by 10×50. To obtain the matching cost between the sample points of a symbol drawing and those of another, we used a mixture of vMDs [2] as a local descriptor of shape because this mixture is invariant with respect to stroke order, number of strokes, and shape scale. The mixture of vMDs was specified by a concentration parameter denoted by κ, and fully tuned for each sampling method. As a result, it was set to $\kappa = 20$ for the four sampling methods. We used the symmetric information divergence between the mixtures of vMDs as the matching cost. Further details on the matching cost calculation using the mixture of vMDs is available in [2].

3.2 Binary Images of Symbols

(1) Dataset

We used a washing tag symbol dataset available from [25], [26]. The dataset consists of 63 binary images of Japanese washing tag symbols, each of which is classified into one of 13 classes according to the meaning of the washing tag symbol. We obtained the symbol contours through a thinning operation on the binary images. Figure 5 shows all the original binary symbol images. In contrast to the time-series dataset described in Sect. 3.1, this dataset does not contain information about the stroke order and number of strokes. Hence, DP detection and vertex detection using PLA requires assigning an artificial order to the contour points. As stated in Sect. 2, our method does not require ordering the points. For this dataset, we examined how the sampling methods processed the thinned contours extracted from the binary images. We set the number of contour points and the space-controlling factor for each symbol drawing to $N = 150$ and $a = 0.15$, respectively. We set the edge width to $w = 3$ pixels.

(2) Evaluation

We selected $n = 18$ sample points for each symbol image using each sampling method. In DP detection, we set the bending-value threshold to $\varepsilon = 3$ so that the resultant number of sample points was around 18. Again, we evaluated

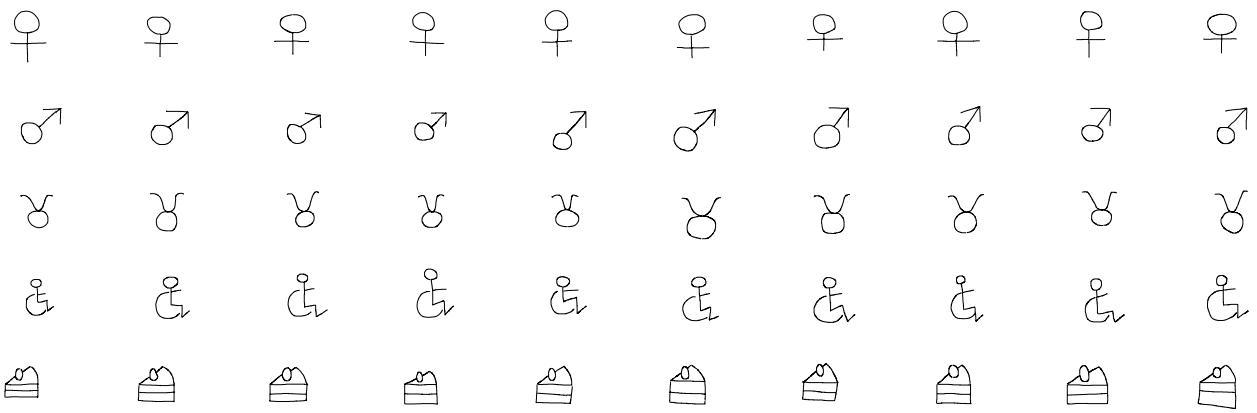


Fig. 4 All the images in the five shape classes of the time-series handwritten drawing dataset: feminine, masculine, Taurus, wheelchair, and cake.

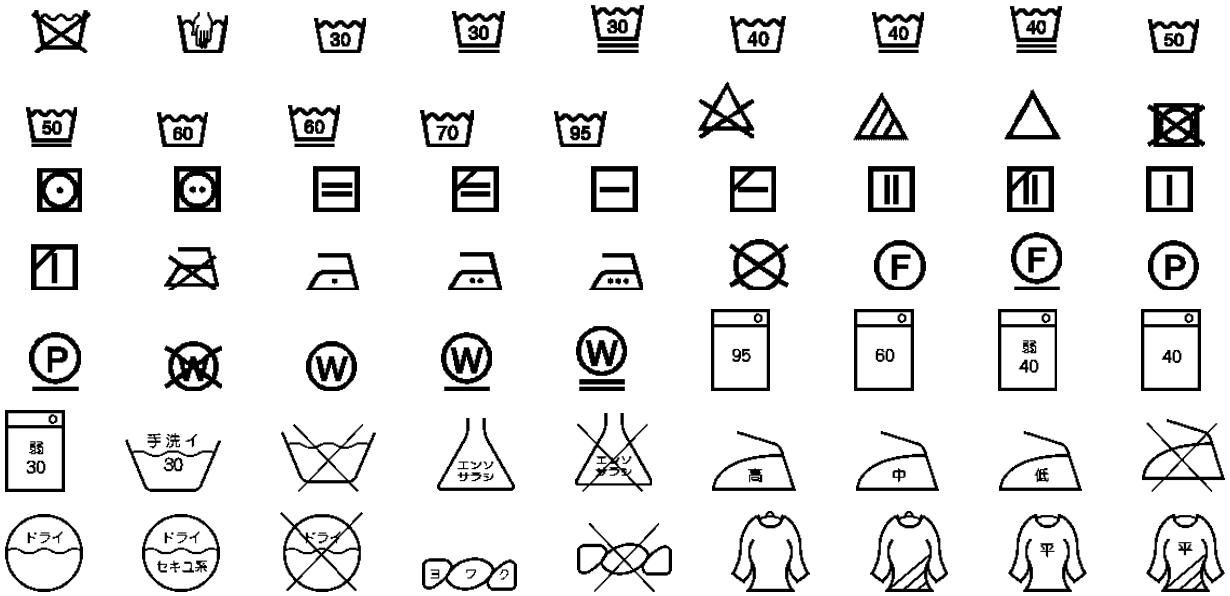


Fig. 5 All the binary images in [25], [26], starting in order from top left to bottom right: 14 new washing, three bleaching, three tumble drying, eight natural drying, four iron-finishing, five new dry cleaning, four wet cleaning, seven old washing, two chlorine bleaching, four ironing, three old dry cleaning, two squeezing, and four drying symbols.

the configuration of the selected sample points using a visual check and retrieval rate. The retrieval rate was computed according to the bullseye test. In this test, we used each symbol image as a query to match its sample points against all the symbol images. For a symbol image, we counted the top m matches with respect to the same local shape descriptor method described in Sect. 3.1, where m denotes the size of the corresponding class. For example, for the "new washing" symbol, we counted the top 14 matches because there were 14 "new washing" symbols in the dataset. The total number of correct matches was at most 433 ($= 14^2+3^2+3^2+8^2+4^2+5^2+4^2+7^2+2^2+4^2+3^2+2^2+4^2$) when all the symbol images were used in turn as queries. Thus, we obtained the overall retrieval rate for these top matches by dividing the number of correct matches by 433. Again, we used a mixture of vMDs as a local descriptor of shape. We set the concentration parameter to $\kappa = 40$ for the uniform sampling method and $\kappa = 30$ for DP detection, PLA vertex detection, and our sampling method.

3.3 Binary Images of Objects

(1) Dataset

We used an image dataset created for experimental psychology [27]. This dataset is a collection of unclassified images. We used four images from the dataset that are shown in Fig. 6. We obtained these images' contours through a thinning operation on the binary images. This dataset differs from the two symbol datasets in Sects. 3.1 and 3.2 because its images are more complex, and they pose a challenge to sampling methods. As with the binary symbol images, we assigned an artificial order to the points on each contour

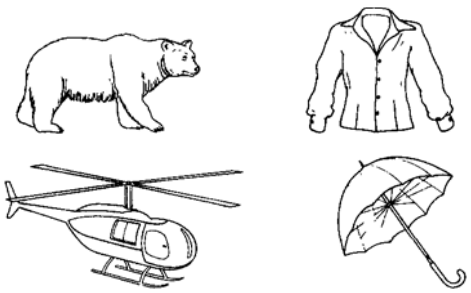


Fig. 6 Four images from the dataset in [27].

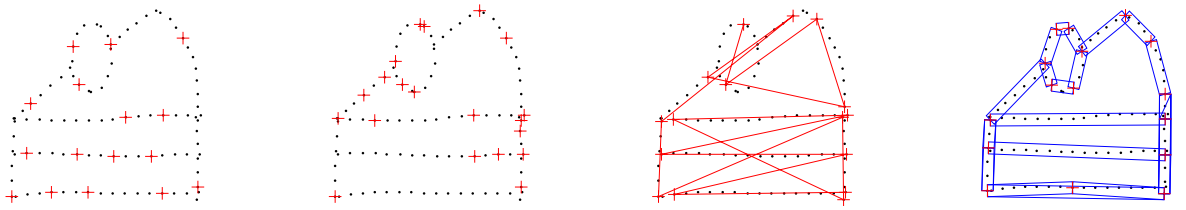
to enable DP and PLA vertex detections. For this dataset, we examined how the sampling methods processed complex contours. We set the number of contour points and the space-controlling factor to $N = 250$ and $a = 0.15$, respectively, for each image. We set the edge width to $w = 8$ pixels.

(2) Evaluation

We selected $n = 25$ sample points on each image using each sampling method. In DP detection, we set the bending-value threshold to $\varepsilon = 52$ so that the resultant number of sample points was around 25. We evaluated the location of the sample points using a visual check only because these images were not classified.

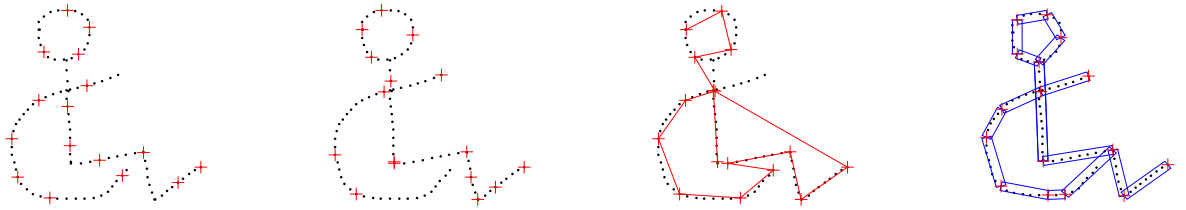
3.4 Results

As space is limited, we present some representative examples of the sample points selected by the four sampling methods in Figs. 7–14. In the figures, a dot and plus sign represent a contour point and sample point, respectively.



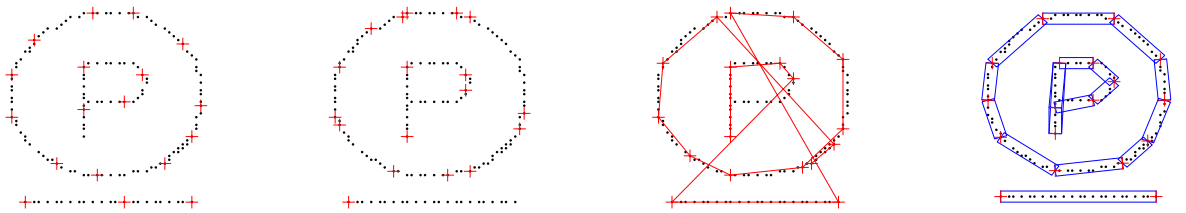
(a) Uniform sampling (b) DP detection (c) Vertex detection by PLA (d) Our sampling method

Fig. 7 Example of a configuration of sample points for a line drawing of a cake.



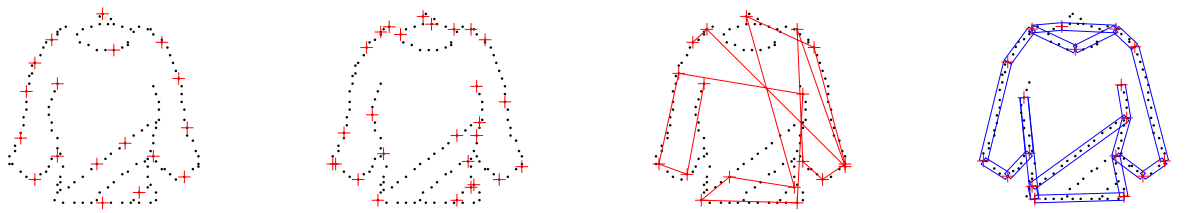
(a) Uniform sampling (b) DP detection (c) Vertex detection by PLA (d) Our sampling method

Fig. 8 Example of a configuration of sample points for a line drawing of a wheelchair.



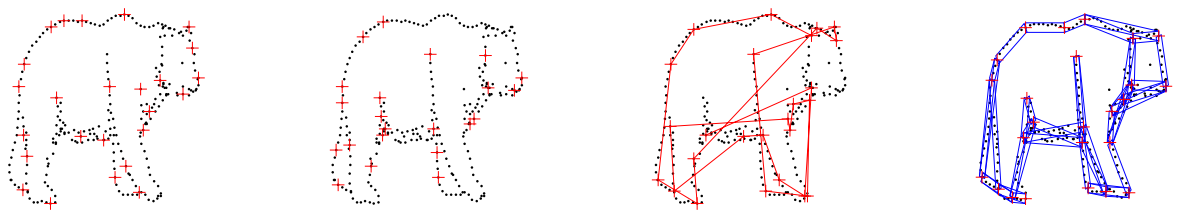
(a) Uniform sampling (b) DP detection (c) Vertex detection by PLA (d) Our sampling method

Fig. 9 Example of a configuration of sample points for a new dry cleaning symbol.



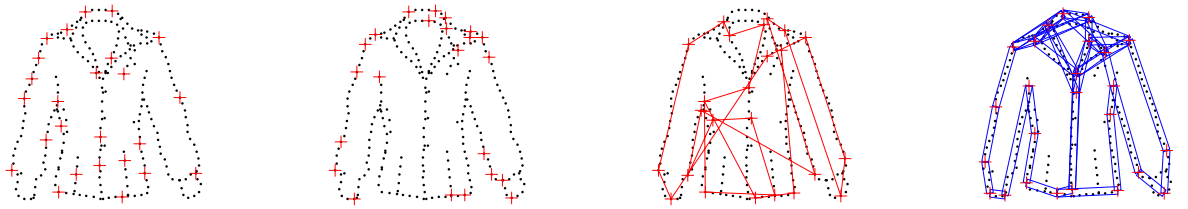
(a) Uniform sampling (b) DP detection (c) Vertex detection by PLA (d) Our sampling method

Fig. 10 Example of a configuration of sample points for a drying symbol.



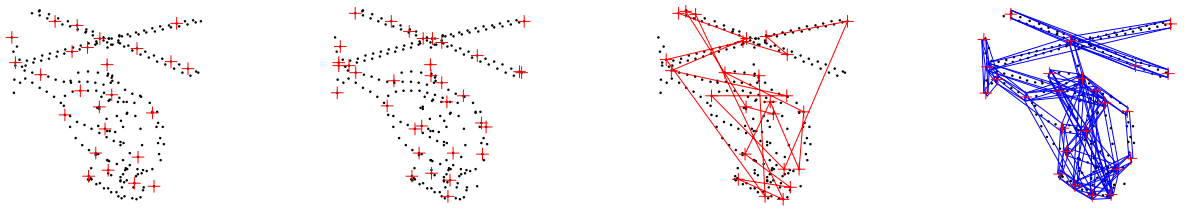
(a) Uniform sampling (b) DP detection (c) Vertex detection by PLA (d) Our sampling method

Fig. 11 Example of a configuration of sample points for an image of a bear.



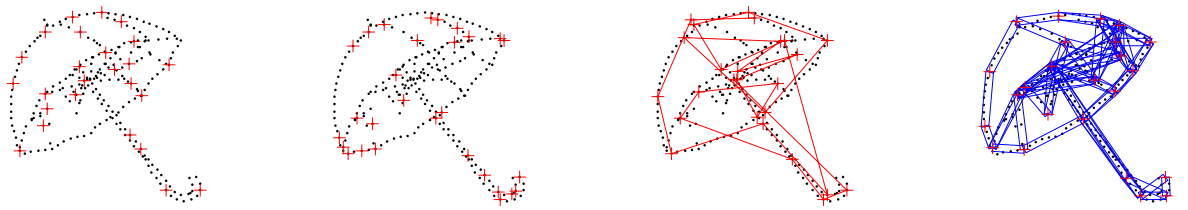
(a) Uniform sampling (b) DP detection (c) Vertex detection by PLA (d) Our sampling method

Fig. 12 Example of a configuration of sample points for an image of a shirt.



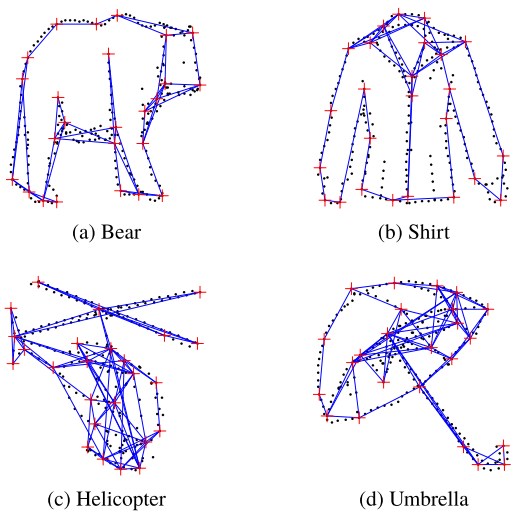
(a) Uniform sampling (b) DP detection (c) Vertex detection by PLA (d) Our sampling method

Fig. 13 Example of a configuration of sample points for an image of a helicopter.



(a) Uniform sampling (b) DP detection (c) Vertex detection by PLA (d) Our sampling method

Fig. 14 Example of a configuration of sample points for an image of an umbrella.



(a) Bear (b) Shirt

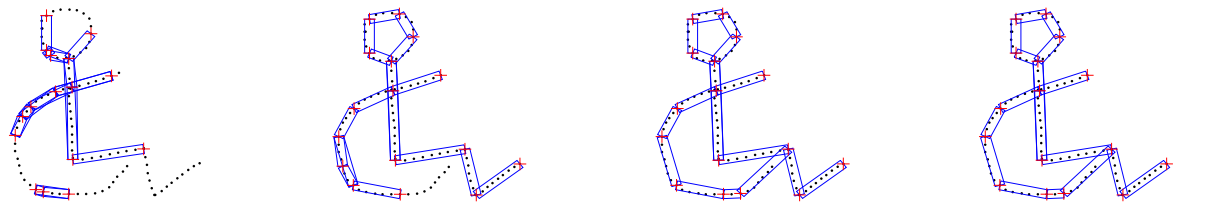
(c) Helicopter (d) Umbrella

Fig. 15 Line segment version of a configuration of sample points using our sampling method.

For reference, we show the corresponding polygonal lines and rectangular edges for vertex detection using PLA and our method, respectively. Regarding the configurations of our method in Figs. 11–14, we also add their line segment versions in Fig. 15 to make the configurations easy to see. By visually inspecting the figures, we observe that, compared with the other methods, our method consistently configured sample points at the end and branch points of contours, such that the rectangular edge graph spanned by the sample points effectively expressed the shape of the contours. In particular, in Fig. 7, we observe that our method selected sample points at the endpoints of each cake layer, and in Fig. 8, we observe that our method selected samples points at the endpoints of the hand, knee, and toe. We provide Fig. 16 to visually check how our method updated the sample points in Fig. 8 during Step 3.

Our method obtained these advantageous sample point configurations because evaluation function F was designed to attach importance to the degree and edge length of the vertices with respect to the rectangular edge graph. Additionally, as shown in Figs. 11–14, our method selected suitable sampling points, even for complex contours in the dataset of binary images of objects.

For an objective evaluation, Table 1 presents the retrieval rates of each of the sampling methods for the first two datasets. We observe from the table that our method was most effective with respect to shape retrieval for both datasets. In particular, our method demonstrated a substantial improvement over the other methods for binary symbol images that had no stroke order and number of strokes. By contrast, DP detection and vertex detection using PLA



(a) Before the updates (Value of F: 51)

(b) After the 1st update (Value of F: 128)

(c) After the 2nd update (Value of F: 143)

(d) After the 3rd update (Value of F: 143)

Fig. 16 Example of updates in Step 3.

Table 1 Retrieval rates for all methods.

Methods	Retrieval rates for handwritten symbol drawings (%)	Retrieval rates for binary symbol images (%)
Uniform sampling method	92.4	69.7
DP detection	75.4	64.7
Vertex detection using PLA	93.4	69.7
Our sampling method	99.6	76.2

were unable to consistently obtain suitable configurations of sample points for the second dataset because of their dependency on the stroke order and number of strokes for the contours.

3.5 Discussion

(1) Dependency on the Location of the Initial Sample Points

As stated in Sect. 2.5, our method relies on the location of the initial sample points (vertices) chosen in Step 2. To alleviate the dependency on the location of the initial sample points, we generated the initial sample points 10 times, and then used the best initial sample points according to the evaluation function. This is a simple but imperfect approach to alleviate the dependency. Generating the initial sample points many times to further alleviate the dependency may entail a high computational cost. Hence, in practice, it is important to set the number of times to generate the initial sample points to achieve a balance between the reduced dependency and the increased computational cost.

(2) Edge Width

We set the edge width w to a larger size for the time-series data of handwritten symbol drawings than for the datasets to consider the pen shake that occurred while drawing the symbol contours. Thus, intuitively, setting the edge width signifies determining an allowable blur (noise) range on the contours.

(3) Resultant Number of Vertices

Table 2 shows the mean and standard deviation (s.d.) of the resultant number of vertices for our method on each dataset. As mentioned in Sect. 2.5, because of Step 5, the resultant number was not always the same as n. In fact, each resultant number was smaller than n. Accordingly, Step 5 worked well at thinning out several vertices that were too close together.

Table 2 Resultant number of vertices.

Stats	Number of vertices for handwritten symbol drawings	Number of vertices for binary symbol images	Number of vertices for binary object images
n	16	18	25
Mean	15.1	16.8	24.7
S.d.	0.85	1.40	0.46

Table 3 Evaluation in terms of PLA.

Methods	Sum of orthogonal distances for handwritten symbol drawings (pixel)
Uniform sampling method	584.0 (373.9)
DP detection	611.2 (385.3)
Vertex detection using PLA	217.8 (193.1)
Our sampling method	739.1 (654.4)

(4) Evaluation in terms of PLA

To examine the quality of our method's configuration in terms of PLA, we composed PLA using the configuration of sample points and calculated the sum of orthogonal distances from the sides of PLA to the contour points, which is a typical measure of fitness of PLA (e.g., see [6], [7]). Because this calculation requires that the contour points are ordered, we used the time-series data of handwritten drawings, each of which had a stroke order and number of strokes. Table 3 shows the mean of the sum of orthogonal distances for each method. The numbers in parentheses are the s.d. of the sums. Clearly, vertex detection using PLA was the best because it was designed to minimize the sum of orthogonal distances. Interestingly, our method was not sufficiently good. This implies that, generally, the fitness of polygonal lines and that of rectangular edge graphs are fairly different.

(5) Limitation

Our method clearly depends on the number of vertices and the edge width. At present, we do not conduct automatic tuning for the parameters affected by the size and fineness of shapes. For example, when some contours are close together and other contours are relatively distant, it may be difficult to find a good edge width. Additionally, the number of vertices appropriate for a shape is application-dependent in practice. Thus, provided we use the rectangular edge graph for shape representation, we have to tune the parameters well using some prior knowledge about the contour shapes.

4. Conclusion

In this paper, we presented an approach to select representative sample points from contours without relying on stroke order and number of strokes. Using a rectangular edge graph, we formulated the sampling problem as a discrete optimization solved using a direct search method. We used different shape datasets to demonstrate that our sampling method configures sample points such that the rectangular edge graph suitably expresses the contours. We showed that our sampling method is more effective with respect to shape retrieval than the comparison sampling methods.

In this work, we set the number of sample points n according to the observed complexity of the shape contours in each dataset. Our future work will include analytically determining the suitable number of sample points to optimize shape representation using the rectangular edge graph.

Acknowledgments

We thank Professor Kazushi Mimura and Professor Tetsuyuki Takahama of Hiroshima City University for their comments on an earlier version of this paper. We also thank Irina Entin, M.Eng., and Maxine Garcia, Ph.D., from Edanz Group (www.edanzediting.com/ac) for editing a draft of this paper. This work was supported in part by JSPS KAKENHI Grant Number JP16K00308 and Hiroshima City University TOKUTEI-KENKYUHI Grant Number 1150347.

References

[1] C. Grigorescu and N. Petkov, "Distance sets for shape filters and shape recognition," IEEE Trans. Image Process., vol.12, no.10, pp.1274–1286, Oct. 2003.
[2] K. Ueaoki, K. Iwata, N. Suematsu, and A. Hayashi, "Matching handwritten line drawings with von Mises distributions," IEICE Trans. Inf. & Syst., vol.E94-D, no.12, pp.2487–2494, Dec. 2011.
[3] H. Fan, Y. Cong, and Y. Tang, "Self-closed partial shape descriptor for shape retrieval," Proc. 19th IEEE International Conference on Image Processing, Orlando, FL, pp.505–508, IEEE, Sept. 2012.
[4] K.V. Mardia and P.E. Jupp, Directional Statistics, Wiley Series in Probability and Statistics, John Wiley & Sons, Chichester, England, 1999.
[5] I.L. Dryden and K.V. Mardia, Statistical Shape Analysis, with Applications in R, 2nd ed., Wiley Series in Probability and Statistics, John Wiley & Sons, Chichester, UK, 2016.
[6] J.-C. Perez and E. Vidal, "An algorithm for the optimum piecewise linear approximation of digitized curves," Proc. 11th IAPR International Conference on Pattern Recognition, vol.III, pp.167–170, IEEE, Aug. 1992.
[7] J.-C. Perez and E. Vidal, "Optimum polygonal approximation of digitized curves," Pattern Recognition Letters, vol.15, no.8, pp.743–750, Aug. 1994.
[8] A.R. Backes and O.M. Bruno, "Polygonal approximation of digital planar curves through vertex betweenness," Information Sciences, vol.222, pp.795–804, Feb. 2013.
[9] E.J. Aguilera-Aguilera, A. Carmona-Poyato, F.J. Madrid-Cuevas, and R. Medina-Carnicer, "The computation of polygonal approximations for 2D contours based on a concavity tree," Journal of Visual Communication and Image Representation, vol.25, no.8, pp.1905–1917, Nov. 2014.
[10] E.J. Aguilera-Aguilera, A. Carmona-Poyato, F.J Madrid-Cuevas, and M.J. Marín-Jiménez, "Fast computation of optimal polygonal approximations of digital planar closed curves," Graphical Models, vol.84, pp.15–27, March 2016.
[11] W.-Y. Wu, "An adaptive method for detecting dominant points," Pattern Recognition, vol.36, no.10, pp.2231–2237, Oct. 2003.
[12] T.P. Nguyen and I. Debled-Rennesson, "A discrete geometry approach for dominant point detection," Pattern Recognition, vol.44, no.1, pp.32–44, Jan. 2011.
[13] M.S. Tahaei, S.N. Hashemi, A. Mohades, and A. Gheibi, "Geometric algorithm for dominant point extraction from shape contour," Pattern Analysis and Applications, vol.17, no.3, pp.481–496, Aug. 2014.
[14] J. Peethambaran, A.D. Parakkat, A. Tagliasacchi, R. Wang, and R. Muthuganapathy, "Incremental labelling of Voronoi vertices for shape reconstruction," Computer Graphics Forum, vol.38, no.1, pp.521–536, March 2019.
[15] M. Cui, J. Femiani, J. Hu, P. Wonka, and A. Razdan, "Curve matching for open 2D curves," Pattern Recognition Letters, vol.30, no.1, pp.1–10, Jan. 2009.
[16] L. Younes, Shapes and Diffeomorphisms, Applied Mathematical Sciences, vol.171, American Mathematical Society, Springer, Berlin, 2010.
[17] J. Pach, ed., Towards a Theory of Geometric Graphs, Contemporary Mathematics, vol.342, American Mathematical Society, Providence, RI, 2004.
[18] R.J. Wilson, Introduction to Graph Theory, 4th ed., Pearson Education, Harlow, Longman, 1996.
[19] A.D. Parakkat, J. Peethambaran, P. Joseph, and R. Muthuganapathy, "A graph-based geometric approach to contour extraction from noisy binary images," Computer-Aided Design and Applications, vol.12, no.4, pp.403–413, 2015.
[20] T.G. Kolda, R.M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," SIAM Review, vol.45, no.3, pp.385–482, 2003.
[21] R.C. Gonzalez, R.E. Woods, and S.L. Eddins, Digital Image Processing Using MATLAB, Pearson Education, Upper Saddle River, NJ, 2004.
[22] K. Sugihara, Mathematics of Graphics, Kyoritsu Shuppan, Tokyo, Japan, Jan. 1995, in Japanese.
[23] A.D. Parakkat, U.B. Pundarikaksha, and R. Muthuganapathy, "A Delaunay triangulation based approach for cleaning rough sketches," Computers & Graphics, vol.74, pp.171–181, Aug. 2018.
[24] K. Ose and K. Iwata, "Data set of line drawings with different stroke order and number," http://www.prl.info.hiroshima-cu.ac.jp/kiwata/oseiwata/, May 2017.
[25] Consumer Affairs Agency, Government of Japan, "Washing tag symbols (before November 30, 2016)," http://www.caa.go.jp/policies/policy/representation/household_goods/guide/wash.html, 2018 (in Japanese).
[26] Consumer Affairs Agency, Government of Japan, "Washing tag symbols (after November 30, 2016)," http://www.caa.go.jp/policies/policy/representation/household_goods/guide/wash_01.

html, 2018 (in Japanese).
[27] J.G. Snodgrass and M. Vanderwart, "A standardized set of 260 pictures: Norms for name agreement, image agreement, familiarity, and visual complexity," Journal of Experimental Psychology: Human Learning and Memory, vol.6, no.2, pp.174–215, March 1980.

Kazuya Ose received B.S. and M.S. degrees from Hiroshima City University, Hiroshima, Japan, in 2016 and 2018, respectively. He was a student member of IEICE until March 2018. He joined Inbility Company Limited in April 2018.

Kazunori Iwata received B.E. and M.E. degrees from Nagoya Institute of Technology, Aichi, Japan, in 2000 and 2002, respectively, and a Ph.D. degree in informatics from Kyoto University, Kyoto, Japan, in 2005. He was a JSPS research fellow from April 2002 to March 2005. He has been with Hiroshima City University, Hiroshima, Japan, since April 2005. His research interests include machine learning, pattern recognition, and information theory. He was an Associate Editor for IEICE Transactions on Information and Systems from July 2013 to June 2017.

Nobuo Suematsu received B.S. and M.S. degrees in physics from Kyushu University, Fukuoka, Japan, in 1988 and 1990, respectively. He received a Ph.D. degree in engineering from Osaka University, Osaka, Japan, in 2000. He joined Fujitsu Laboratories Ltd. in 1990. From April 1994 to December 2018, he was with Hiroshima City University, Hiroshima, Japan.