Applied Soft Computing 68 (2018) 932–943

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# Linking soft computing to art: introduction of efficient k-continuous line drawing

Yanjie Zhou, Gyu M. Lee*

*Department of Industrial Engineering, Pusan National University, Busan, Republic of Korea*

ARTICLE INFO

*Article history:*
Received 18 December 2016
Received in revised form 22 August 2017
Accepted 21 November 2017
Available online 24 November 2017

*Keywords:*
k-Continuous line drawing
Image segmentation
Traveling salesman problem
Weighted voronoi diagram
Ant colony system algorithm
Minkowski distance

ABSTRACT

Continuous line drawing (CLD) is a technique used in the field of art, in which the pen does not leave the paper until the sketch is completed. In this study, a novel technique, k-continuous line drawing (k-CLD), is proposed; this technique enables a visual image to comprise k closed non-intersecting lines. k-CLD involves the following challenges: 1) partitioning the target image into k regions, 2) stippling each region without distorting the target image, and 3) connecting the stippled dots in each region using a single closed, non-intersecting line. This study identifies and implements efficient algorithms to produce high-quality k-CLDs. Further, an improvement to the graph-based image segmentation algorithm has been proposed using the Minkowski distance to evaluate dissimilarity difference and demonstrated its effectiveness to partition the target image into k regions. Next, well-spaced stippled dots were generated in each region using a weighted Voronoi diagram. Finally, the stippled dots were connected by a single non-intersecting line, obtained by solving a traveling salesman problem (TSP) in each region. The meta-heuristic to solve the TSP was an ant colony system algorithm. The proposed methodologies were tested on a wide variety of images to demonstrate their effectiveness and efficiency.



## 1. Introduction

Continuous line drawing (CLD) is a technique used in the field of art, in which images are expressed by a single line. Besides drawing outlines, internal shapes, and sometimes shades, a pen must move back and forth across the surface of the paper, with lines doubling back on each other, so that the drawing becomes a free-flowing and unbroken line. Although CLD is a very powerful technique to generate a creative and impressive artwork, it is difficult to use for both beginners and human artists. In this paper, this CLD, which produces black-and-white sketches, is extended to develop a new artistic technique that can express multi-shaded and color images into high-quality k-continuous line drawings (k-CLDs) while preserving the original image's features and impressions.

The earliest work on single-line CLD [1] described the use of the traveling salesman problem (TSP), which involves finding the minimum-cost tour in a graph, to create a CLD of a target image. Fig. 1(a) shows an example CLD produced by the aforementioned study [1]. Kaplan and Bosch [2] proposed a modified grid-based method and presented an alternative algorithm for city distribution, called the ordered dithering algorithm [3], which yields more attractive line drawings. Finally, they rendered the cities by B-spline control points and a smooth curve. Fig. 1(b) shows an example CLD with the city distribution and rendering styles produced by Kaplan and Bosch [2].

Bosch proposed a new TSP-based method, which describes the construction of tours that wind through the cities such that certain pairs of user-selected, city-free regions occupy one side of the tour, while other regions occupy the opposite sides [4]. Their method converted a black-and-white user-supplied target image into an arrangement of dots (cities of the TSP). Finally, it constructed a group of traveling salesman's tours, as shown in Fig. 2 [4].

Recently, Li and Mould proposed an algorithm for constructing CLDs using the tone and structural information obtained from target images [5]. In their method, users were required to input the skeleton of the target image. The method then dilated the structure by spatially varying dilation parameters. Finally, it generated CLDs by tracing the boundary of the resulting region. Fig. 3 shows an example CLD constructed by their method [5], which is designed to look like a decorated tree.

An image comprises multiple sets of pixels in multiple areas. Each pixel set represents a different object. For example, a portrait consists of the forehead, eyes, nose, lips, chin, and torus, each of which is represented by a set of pixels. This study aims to inves-

* Corresponding author.
*E-mail address:* glee@pusan.ac.kr (G.M. Lee).

https://doi.org/10.1016/j.asoc.2017.11.035
1568-4946/

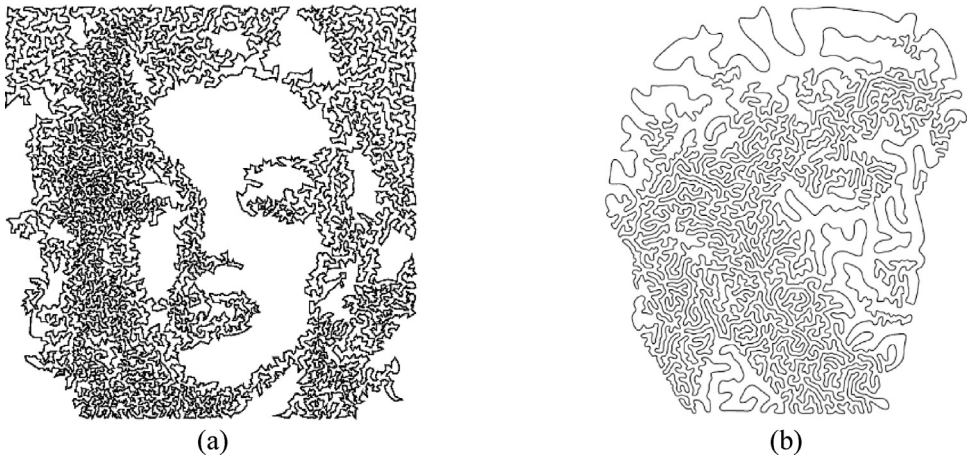


**Fig. 1.** Continuous line drawings (CLDs); (a) Marilyn Monroe (11508 cities) [1] and (b) David (500. cities)[2].

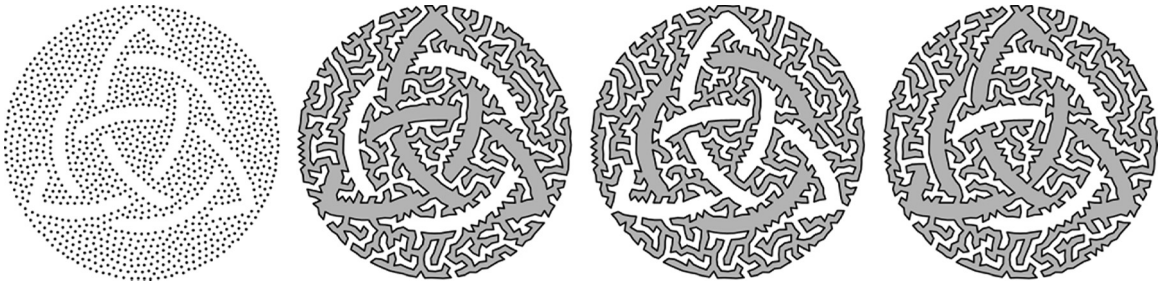

**Fig. 2.** Connecting the dots: the ins and outs of traveling salesman problem (TSP) art [4].

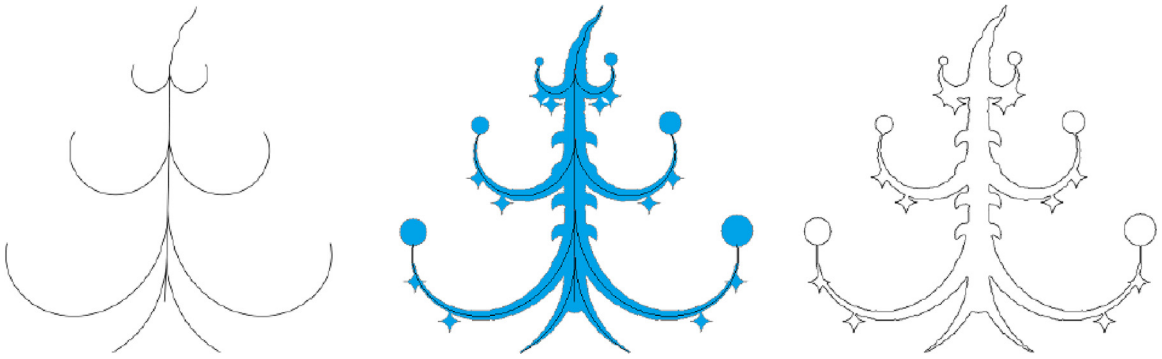

**Fig. 3.** Designing a CLD from an image skeleton [5].

tigate methodologies for converting an image into a k-CLD and devise a software tool that converts a user-supplied image into an art image comprising k-continuous lines without losing the original image's features.

The proposed methodologies are implemented by three algorithms. First, we partition the target image into k segments (or regions). Second, we stipple the target image in each segment with dots on a blank canvas. Finally, the dots in each segment are considered as a set of cities (or nodes) that must be visited by a traveling salesman while forming a Hamiltonian circuit. The distance between any pair of nodes is calculated using the Euclidean distance. Experiments have confirmed that tours close to Hamiltonian circuits tend to be non-intersecting. These three algorithms in the proposed methodologies convert the target image into a k-CLD.

The remainder of this study is organized as follows. The following section introduces related literature on image segmentation, stippling, and TSP solving. Section 3 introduces the preliminaries and provides detailed explanations of the proposed methodologies. Section 4 summarizes the computational results. Section 5 provides the conclusions of the study.

## 2. Related work

The purpose of this study is to link optimization approaches to a new type of artistic technique. To this end, it investigates a novel procedure that converts a target image into a kCLD and proposes a series of algorithms for implementing this procedure. These algorithms sequentially perform image segmentation, stippling, and TSP-solving.

Image segmentation is a fundamental problem or algorithm in computer vision. It divides an image into several disjoint regions, each corresponding to a meaningful part of the image. Segmentation has been extensively researched over the last thirty years and is implemented by various methods including the threshold method [6], color-based segmentation, transform method [7], and texture method [8]. Methodologies related to those of the

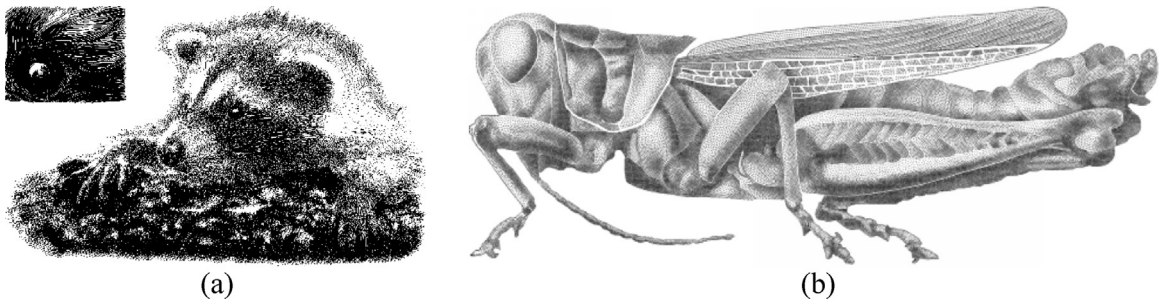


**Fig. 4.** (a) Raccoon with details (inset) of the stroke character [17] and (b) grasshopper generated from 60,000 dots [18].

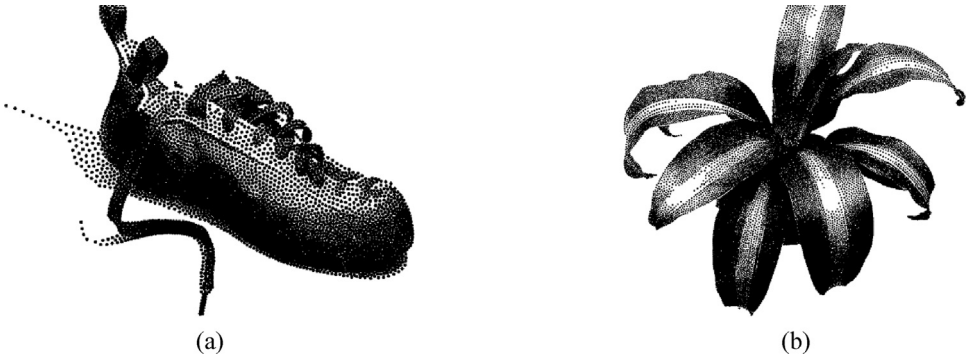


**Fig. 5.** (a) Climbing shoe with 5000 dots of radius $3 \times 10^{-3}$(m) and (b) corn plant with 20,000 dots of radius $1.5 \times 10^{-3}$(m) [19].

present study were proposed by Shi and Malik [9] and Felzenszwalb and Huttenlocher [10]. Shi and Malik treated image segmentation as a graph partitioning problem and introduced a novel global criterion called normalized cuts, which measures the total dissimilarity among different groups and the total similarity within each group [9]. Although normalized cuts are formulated as a graph partitioning problem, the actual approximation is optimized via a non-combinatorial method. Felzenszwalb and Huttenlocher defined a predicate method to determine whether a boundary exists between two regions in a graph-based representation of the image [10]. Their algorithm is implemented in near-linear time to the number of graph edges and is practically quick. Importantly, their method preserved the details in low-variability regions of the image while ignoring them in high-variability regions.

Stippling uses small dots to create patterns that simulate varying degrees of solidity or shading. This technique emerged when image printing was newly introduced to the printing industry [11]. Originally, printers mechanically applied the image halftoning technique, which approximated the original image with a limited number of intensity levels, typically black and white [12]. Today's printers are more advanced and employ more automated dithering algorithms that can randomize the dot patterns, create a more natural appearance, and produce realistic images using far less ink than fully saturated ones [13,14].

Lloyd's algorithm finds the evenly spaced sets of points in subsets of Euclidean spaces [15]. It was introduced by McCool and Fiume to the computer graphics area for generating sampling point sets [16]. Salisbury et al. [17] proposed a pen-and-ink illustration technique that can produce stipple illustrations when pen strokes are replaced with dots. Fig. 4(a) shows an image of raccoon created using the stroke characteristics of Salisbury et al.'s algorithm [17]. Deussen et al. [18] presented a stipple-drawing method that renders polygonal models into a continuous-tone image and then roughly places the stippled dots by a dithering algorithm applied on the target image. After that, to settle these roughly spaced dots to their definite position they used the Lloyd's algorithm. Fig. 4(b) shows a grasshopper with 60,000 dots constructed using Deussen et al.'s algorithm [18].

Secord presented two non-interactive techniques that can generate stipple drawings by constructing weighted Voronoi diagrams from grayscale images [19]. Secord's iterative technique converts the input images into high-quality stipple drawings by precomputing the dot distributions. Fig. 5(a) and (b) show a climbing shoe and a corn plant constructed from 5000 and 20,000 dots by Secord's algorithm [19], respectively.

Kim et al. [20] presented a new stippling style, in which the stippled dots collectively follow the direction of the nearest image feature. Their algorithm can automatically produce stippled renderings from target images, following the style of professional hedcut illustrations. They proposed a novel dot placement algorithm that adapts the sizes of the stipple dots to suit the local shapes. Fig. 6 shows two results of Kim et al.'s algorithm [20].

As described above, there have been many efforts to express images in various ways by using mathematical and computer algorithms. The present study intends to propose an interesting approach based on metaheuristic to generate art image. TSP is one of the most famous problems in combinatorial mathematics and has commanded considerable attention from mathematicians and computer scientists. In graph theory, TSP identifies the most efficient Hamiltonian circuit through which a traveling salesman can visit $n$ cities at the lowest cost. The only constraint is that the traveling salesman must visit each city exactly once before returning to the original departure city [21].

## 3. Preliminaries

In this section some basic concepts and methodologies related to CLD has been presented and discussed for the readers to understand the proposed k-CLD better. Fig. 7 shows how the proposed procedures split a target image into smaller images and convert each of them to the corresponding CLDs.

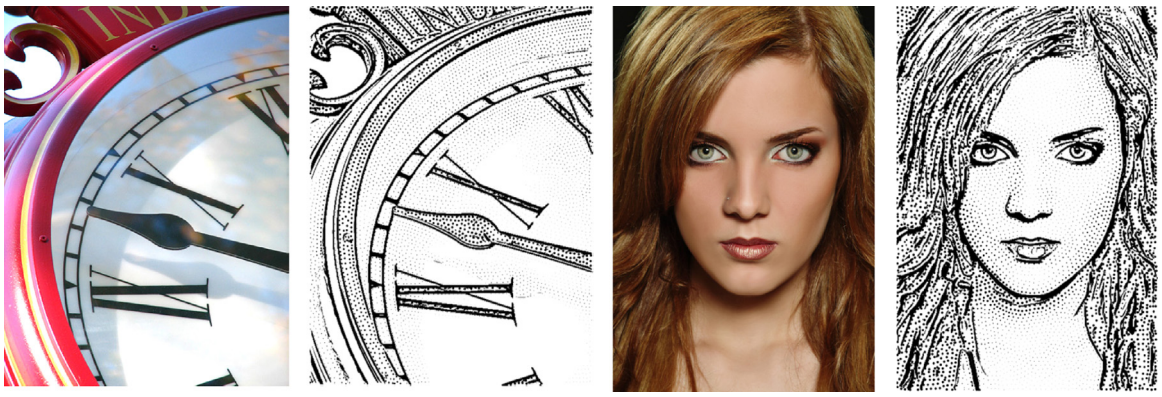

**Fig. 6.** Feature-guided image stippling [20].

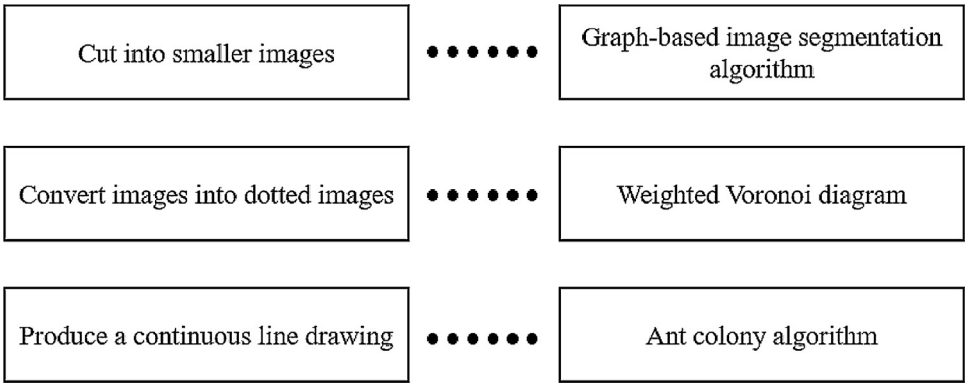


**Fig. 7.** Preliminary overview of the proposed methodologies.

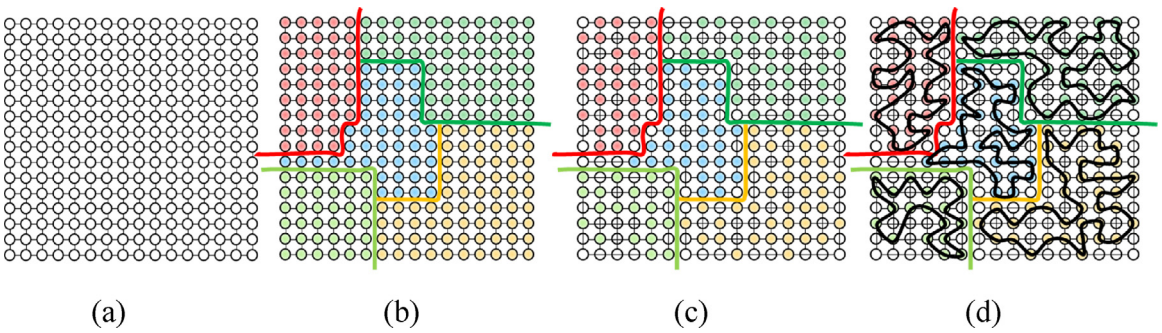


**Fig. 8.** Procedures of the proposed methodologies; (a) user-supplied image, (b) segmentation, (c) stippling, and (d) TSP tours.

Fig. 8 demonstrates these procedures intuitively. In the user-supplied image (Fig. 8(a)), rectangular RGB pixels are assumed at each $(x, y)$ coordinate. Each small blank circle in Fig. 8 represents one pixel. The color of each pixel is expressed as a three-integer tuple (R, G, B), where $0 \leq R, G, B \leq 255$. Changing the RGB values of the pixels alters the tone, color, shape, and inspiration of the image. Based on the differences among and changes in the RGB values, the image is divided into k regions (Fig. 8(b)). Note that the pixels with similar RGB values tend to cluster in the same region. If a considerable difference in RGB values exists between two neighboring pixels, both pixels are located on the boundary. In Fig. 8(b), different regions are represented as clusters of differently colored circles. Note that the colored curves are imaginary; they merely demonstrate how the image is separated into regions. Each region in Fig. 8(b) comprises a certain number of colored circles (pixels) determined by the segmentation algorithm. Each region is then stippled with colored and blank circles. This process converts a portion of the colored circles in the region into blank ones without losing the features and impression of the original images (see Fig. 8(c)). The remaining pixels need to be connected via a continuous line. The tour of a line can be generated in various ways. Although densely connected and crossed pictures are often desired, we are interested in sparse line drawings with few line crossings. These sparse line drawings are conjectured to be Hamiltonian circuits or good CLDs with relatively short lengths.

### 3.1. Image segmentation

Image segmentation is the process of partitioning an image into multiple regions. Segmentation has been extensively researched over the last thirty years. This study adopt and extends the graph-based segmentation algorithm proposed by Felzenszwalb and Huttenlocher [10]. To increase the effectiveness of the graph-based segmentation algorithm, we introduce the preprocessing algorithm

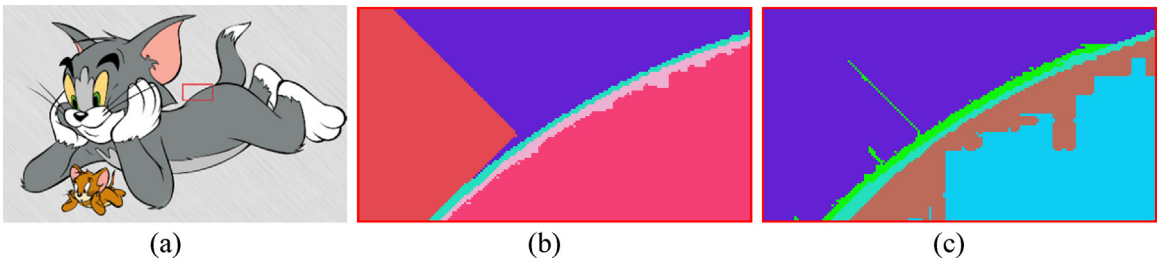


**Fig. 9.** Crude segmentation results of the "Tom and Jerry" image.

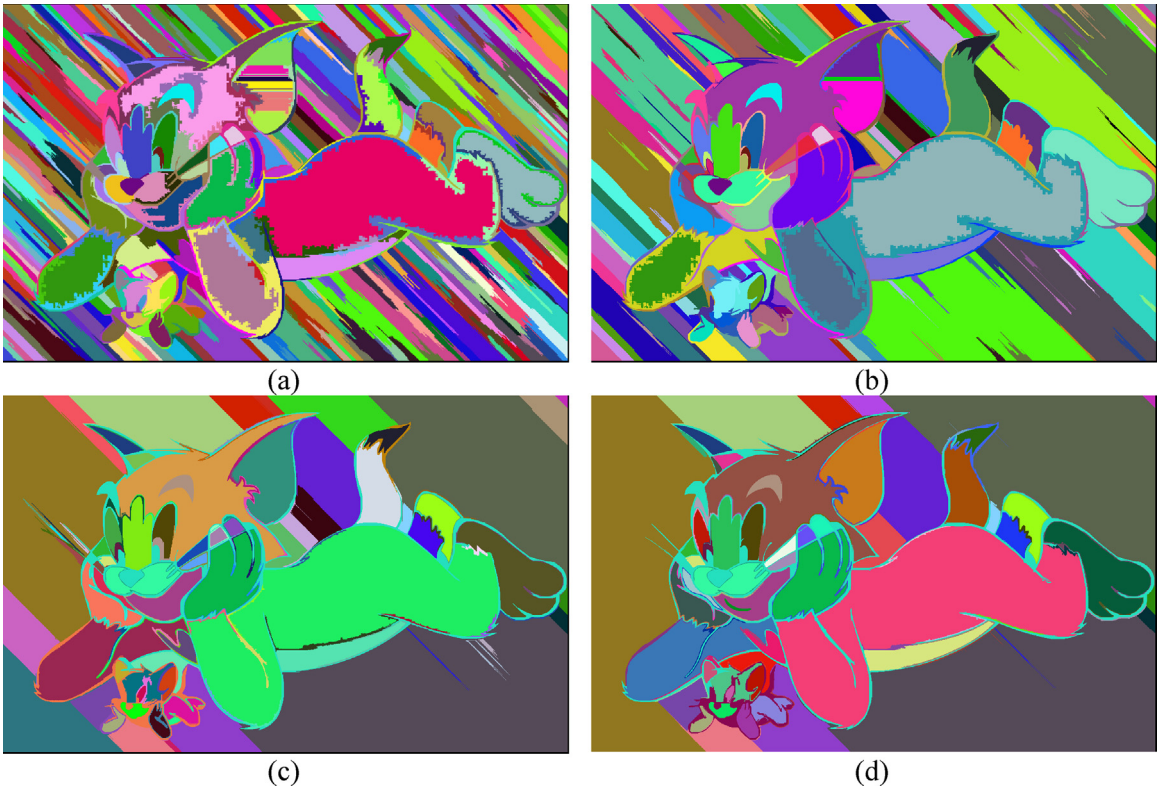


**Fig. 10.** Segmentation results of different norm $p$ values in the Minkowski distance, (a) $p = 0.33$, (b) $p = 0.5$, (c) $p = 2$ and (d) $p = 4$.

of Xu *et al.* [22], which sharpens the major edges by increasing the steepness of the transition while eliminating a manageable number of low-amplitude structures. Our additional preprocessing operation helps keep pixels having similar RGB values within a cluster.

Before introducing our improvements to the existing algorithm, we outline the basics of the graph-based segmentation algorithm [10]. Let $V$ be the set of pixels on an input image $I$. Let $v_i\left(x^i, y^i\right)$ be a pixel on image $I$, where $v_i \varepsilon V$ and $x^i$ and $y^i$ are the horizontal and vertical coordinates, respectively. The input image $I$ can be considered as a graph $G = (V, E)$, where $E$ is the set of undirected horizontal or vertical edges between two neighboring pixels. The output is a segmentation of $V$ into regions $S = (C_1, \cdots, C_r)$, where $C_1 \cup \cdots \cup C_r = V$ and $C_i \cap C_j = \emptyset$ for $i \neq j$. The dissimilarity between neighboring pixels $v_i$ and $v_j$ is defined as a non-negative function $w\left(v_i, v_j\right)$ given by

$$w\left(v_i, v_j\right) = \sqrt[2]{\alpha\left(R(v_i) - R\left(v_j\right)\right)^2 + \beta\left(G(v_i) - G\left(v_j\right)\right)^2 + \gamma\left(B(v_i) - B\left(v_j\right)\right)^2} \tag{1}$$

where $R(x, y)$, $G(x, y)$, and $B(x, y)$ are the RGB values of pixel at coordinate $(x, y)$.

Graph-based image segmentation is based on a straightforward rule that if neighboring pixels have a good similarity, i.e., similar RGB values, they must clustered within the same region. Starting from a single pixel, neighboring pixels with similar RGB values are merged into the same segmentation as the process continues.

The steps to identify the boundaries of the pre-determined number of regions (segments) are explained in the following. If the external difference between two regions exceeds the internal differences of both regions by $\tau$, where $\tau$ is an adjustable threshold function, the two regions are separated by a boundary. $\tau$ is defined as;

$$\tau(C) = k/|C|, \tag{2}$$

where $|C|$ denotes the cardinality of $C$, and $k$ is a user-controlled parameter. A large $k$ favors the larger segments.

$Int(C)$ is the internal difference function of region $C$. Its value is the maximum weight of the edges in a graph comprising region $C$ and a horizontal and vertical edge set $E$:

$$Int(C) = \max_{\left(v_i, v_j\right) \in MST(C,E)} w\left(v_i, v_j\right), \tag{3}$$

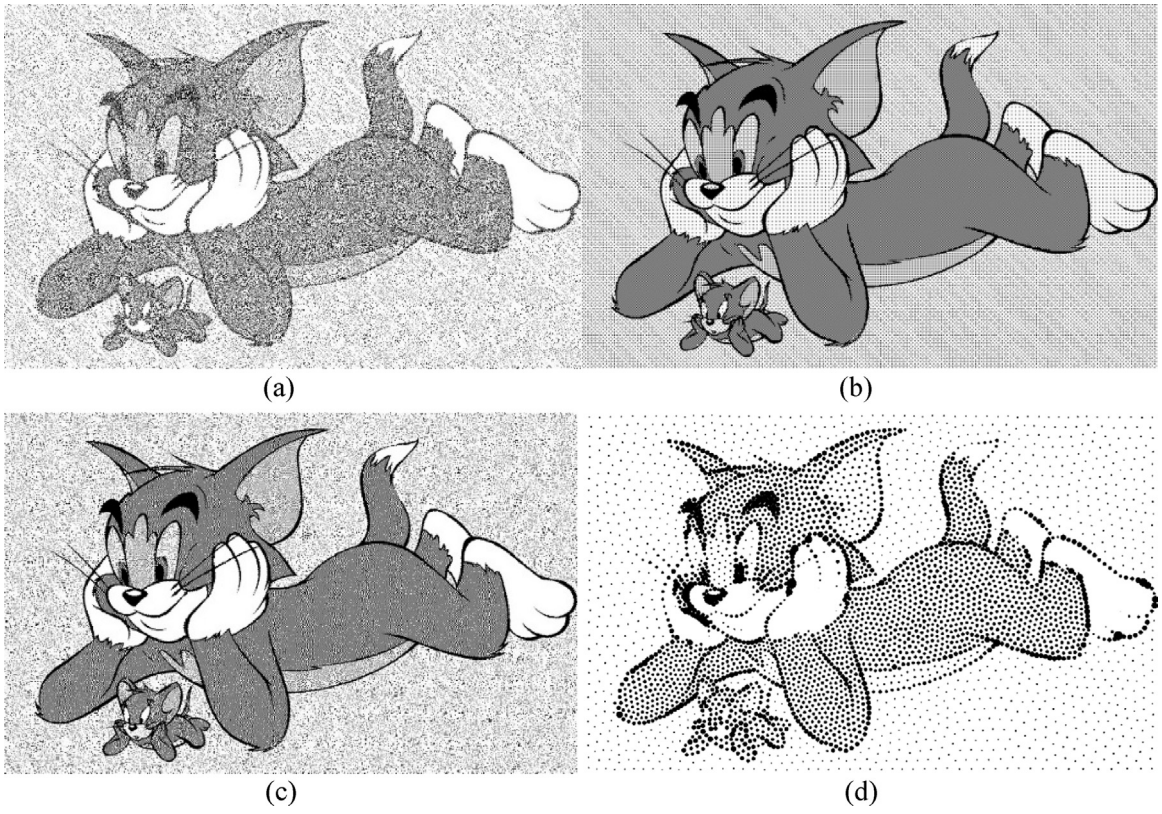


**Fig. 11.** Results of stippling methods (a) Bosch and Herman, (b) ordered dithering, (c) Floyd–Steinberg dithering, and (d) weighted Voronoi diagram.

**Modified Lloyd's iterative algorithm**

**for** all Voronoi polygon $R_i$
  **while** generator $q_i$ is not on centroid $A_i$
    calculate the density function $\rho(q_i)$
    calculate the density function $\rho_{weight}(q_i)$
    calculate the mass centroid $A_i^{weight}$
    set $q_i$ to $A_i^{weight}$
  **endwhile**

**Fig. 12.** Modified Lloyd's iterative algorithm.

where $MST(C, E)$ is the minimum spanning tree. $MInt(C_1, C_2)$ is the minimum internal difference function and is defined as

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)). \quad (4)$$

The difference function $Diff(C_1, C_2)$ considers the minimum weight among all edges connecting any pair of pixels in two regions $C_1$ and $C_2$, where $C_1, C_2 \subseteq V$ and $v_i \in C_1$, $v_j \in C_2$. This function is defined as

$$Diff(C_1, C_2) = \begin{cases} \infty, & \text{if there is no edge} \\ \min\limits_{(v_i, v_j) \in E} w(v_i, v_j), & \text{otherwise.} \end{cases} \quad (5)$$

Finally, the pairwise-comparison predicate function $D(C_1, C_2)$ is defined as

$$D(C_1, C_2) = \begin{cases} \text{true}, & \text{if } Diff(C_1, C_2) > \text{MInt}(C_1, C_2) \\ \text{false}, & \text{Otherwise} \end{cases}. \quad (6)$$

This function determines the existence of a boundary between two regions.

Smooth boundaries in the segmented images are problematic in our implementation. However, the basic implementation of [10] generates unnecessary spikes and blocks near the boundaries of certain target images.

The present study proposes a new dissimilarity function defined as follows:

$$w(v_i, v_j) = \left(\alpha|R(v_i) - R(v_j)|^p + \beta|G(v_i) - G(v_j)|^p + \gamma|B(v_i) - B(v_j)|^p\right)^{\frac{1}{p}} \quad (7)$$

where $R(x, y)$, $G(x, y)$, and $B(x, y)$ are the RGB values of the pixel at coordinate $(x, y)$. The constants $\alpha$, $\beta$, and $\gamma$ are weight parameters that control the segmentation quality. More specifically, these weights control the color tone of the target images, producing better segmentations and their boundaries. Excluding those three constants, the dissimilarity function can be defined using the Minkowski distance.

The Minkowski distance is defined as

$$d(x_i, y_j) = \left(\sum_{i=1}^{n} |x_i - y_j|^p\right)^{1/p}, \quad (8)$$

where $p$ is the p-norm.

The distance of order p between two points is called the Minkowski distance of p (p-norm distance). In [10], the Euclidian distance is the 2-norm distance in three-dimensional space, a generalization of the Pythagorean theorem. The 1-norm distance is the distance between two points in a city configured as a square block with no one-way streets and is hence called the Manhattan distance or taxicab norm.

Fig. 9(b) and (c) show the segmentation result of a part of the "Tom and Jerry" image marked by a rectangle in Fig. 9(a). To perform well over a wide spectrum of user images, the segmentation algorithm must be sufficiently robust. Therefore, the weight parameters and Minkowski distance in the dissimilarity function has been intensively studied. Fig. 9(a) is the input image, and Fig. 9(b) and (c) are the segmentation results of the image in rectangle using $p$

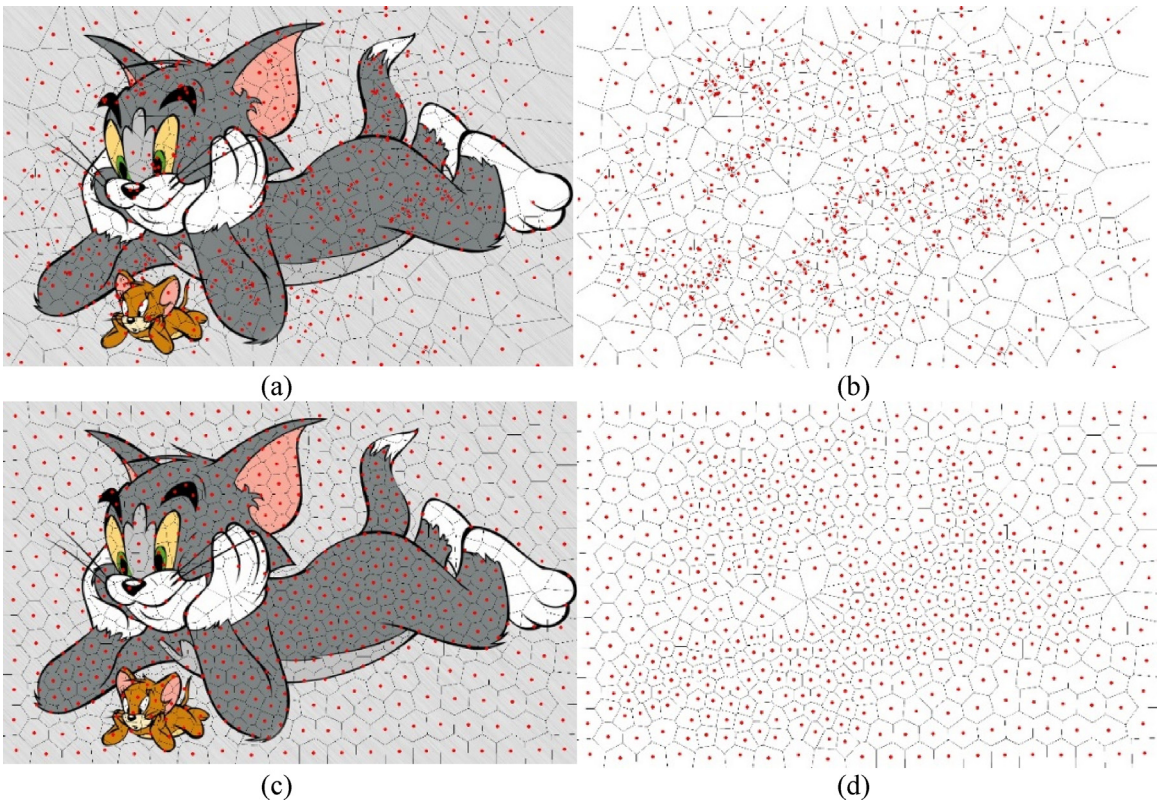


**Fig. 13.** Voronoi diagram (a) with and (b) without the original image, and weighted Voronoi diagrams (c) with and (d) without the original image.

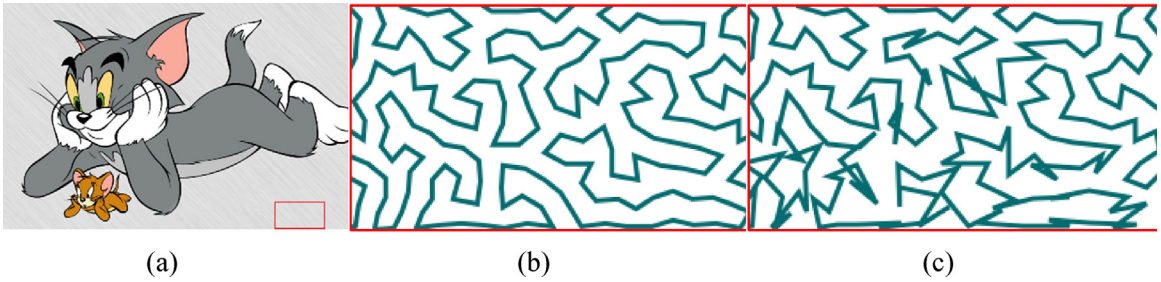


**Fig. 14.** Difference in the image quality caused by the number of ants on the input images; (a) the original image marked by a rectangle for Figs. (b) and (c), (b) image quality created employing 100 ants, and (c) image quality created employing 10 ants.

set to 4 and 8, respectively. Fig. 9(c) is punctuated by sharp corners compared to Fig. 9(b) which has a relatively smooth boundary.

Fig. 10 compares the segmentation results when the graph-based segmentation algorithm is used with the Minkowski distance as the dissimilarity difference. In these segmentations, norm $p$ was set to 0.33 (Fig. 10(a)), 0.5 (Fig. 10(b)), 2 (Fig. 10(c)) and 4 (Fig. 10(d)), respectively. As shown, the smooth boundaries and segmentation were obtained in Fig. 10(d) when using $p = 4$ in the Minkowski distance. However, experimenting with multiple instances shows that the appropriate $p$ value depends on the input image. An algorithm to determine the best $p$ values for different input images can be studied in the future.

As mentioned above, image segmentation is the first step of k-CLD. Therefore, the image segmentation has a great impact on the k-CLD quality. In the present study, the norm $p$ was identified as a useful control parameter for achieving high-quality segmentations with smooth boundaries.

### 3.2. Stippling

Despite the large amount of research on stippling methods, Bosch and Herman's grid-based algorithm [1] remains as one of the most popular stippling methods. This algorithm resizes the target image to a pixel map with $km$ rows and $kn$ columns, where $k$ is an integer number. The pixel map is then converted to grayscale format, and the target image is partitioned into $m$ rows and $n$ columns of $k \times k$ square. Cities are randomly placed in each square, depending on the darkness value of the square. $\mu_{ij}$ is the mean grayscale value of the pixels in square $(i, j)$, where $i$ and $j$ denote the row and column numbers, respectively. The average darkness $g_{ij}$ of square $(i,j)$ varies from 0 (completely white) to $\gamma$ (completely black) and is calculated as $g_{ij} = \gamma - \left[\frac{\gamma\mu_{ij}}{256}\right]$. Here $\gamma$ denotes the maximum desired number of pixels in a square. In Bosch and Herman's implementation [1], $g_{ij}$ is the actual number of pixels and the number of cities in square $(i,j)$.

The tone and texture of the images can be expressed in many ways. Another stippling method is halftoning, which reproduces a

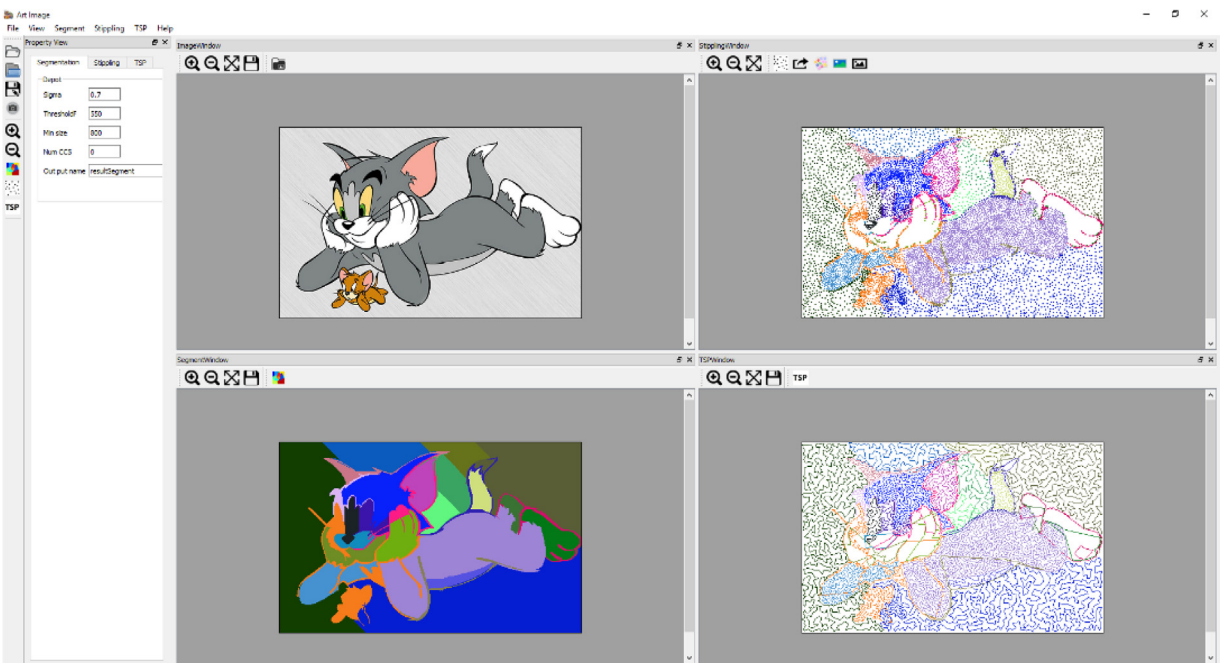

**Fig. 15.** Screenshot of the k-CLD system interface.

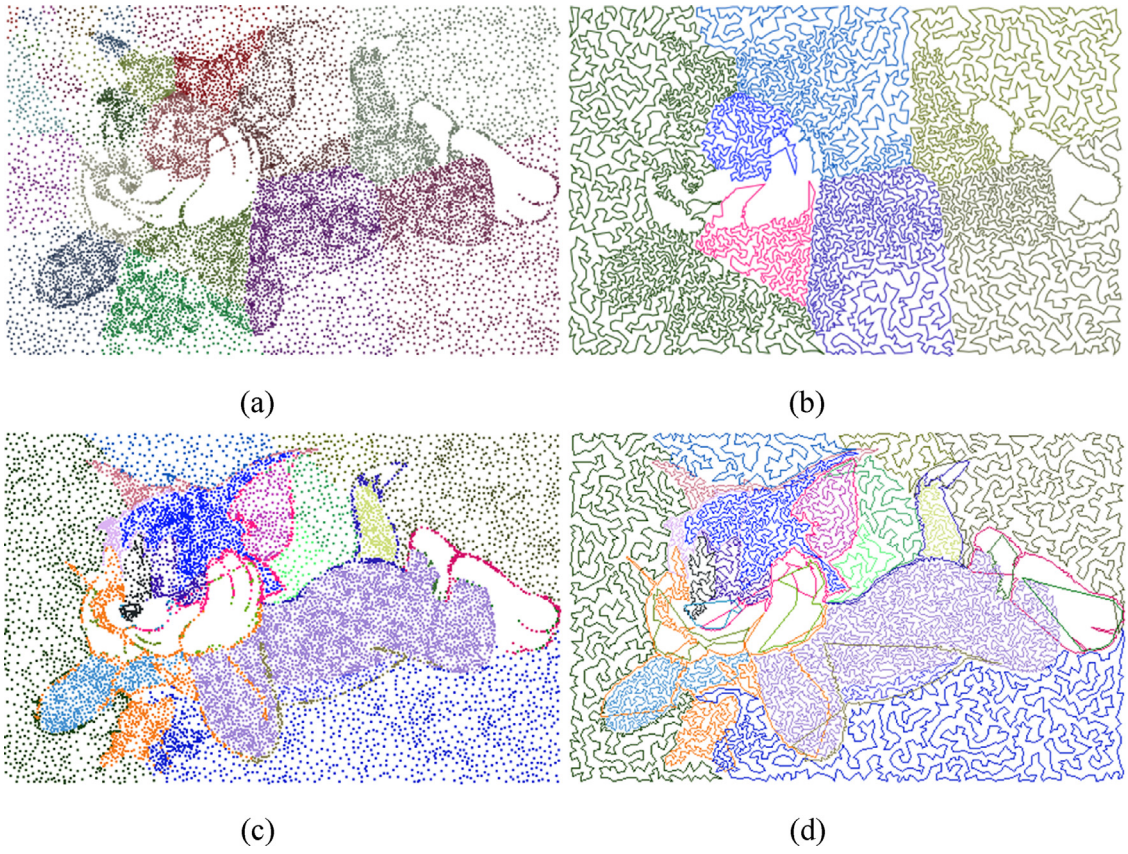


**Fig. 16.** Differences in the image quality of different segmentation algorithms. (a) Stipplings segmented by the k-means clustering algorithm and (b) k-CLD produced from the stipplings in (a); (c) stipplings segmented by the proposed algorithm and (d) k-CLD produced from the stipplings in (c).

continuous-tone grayscale image by varying the sizes of tiny black dots arranged in a regular pattern (i.e., halftone pattern) [23]. This technique has been used in the publishing industry and in printers. Similar to halftoning, digital halftoning decomposes an image into a grid of halftone cells. Digital halftone images are most commonly generated by dithering algorithms such as the ordered dithering [3] and Floyd–Steinberg dithering [24] algorithms. The characteristic of dithering algorithms is that these algorithms create an output image containing as many dots as there are pixels in the input image. In other words, dots of various diameters exist in all grid

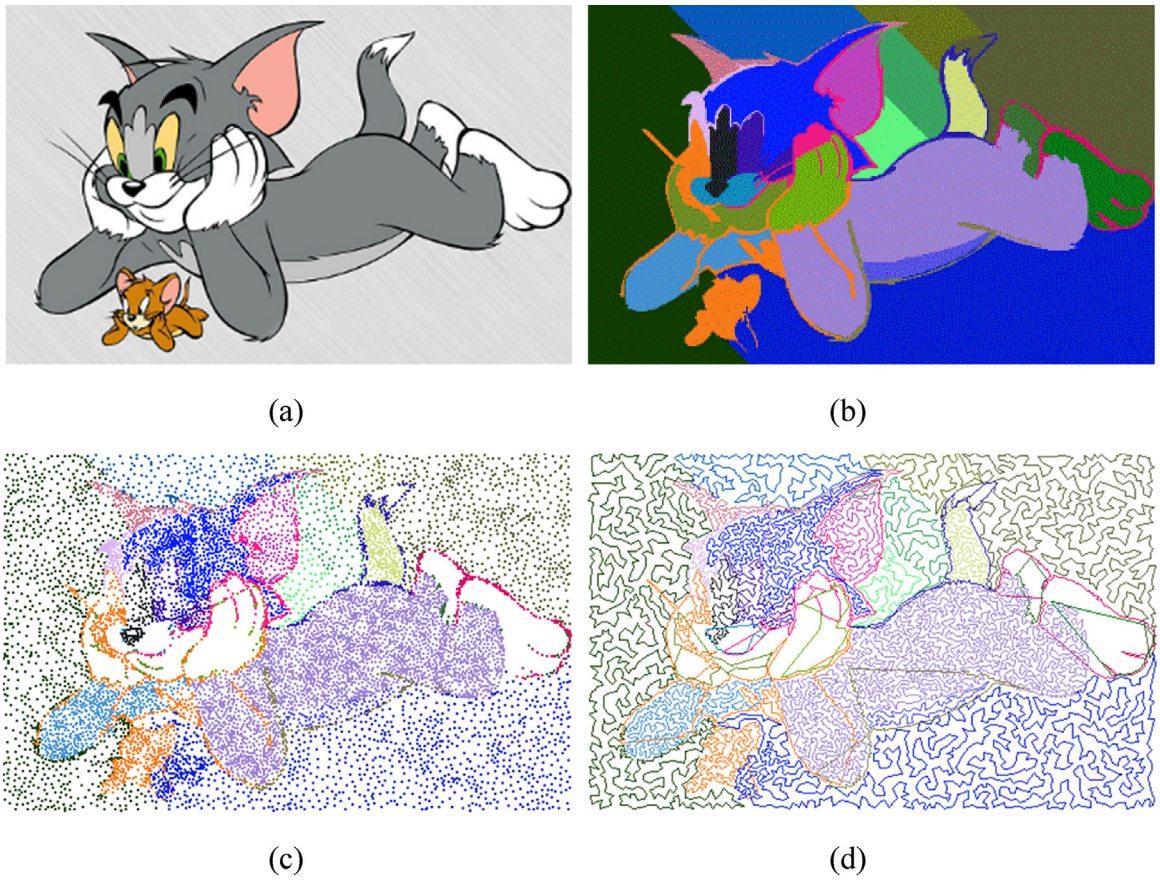


**Fig. 17.** Results of the proposed methodologies: (a) input image, (b) segmentation result, (c) stippling result, and (d) k-CLD result.

locations of the image when halftoning is used. The number of dots in each region is manipulated using a Voronoi diagram.

A Voronoi diagram is generated by choosing $n$ randomly generated points (called generators) among points at all grid locations. Each generator is then contained within a convex polygon (called a Voronoi polygon), such that all points at grid locations within a polygon are closer to its generator than to any other generators. The Voronoi diagram algorithm has linear time complexity and can be implemented in $O(n \log n)$ time. As the generators in a Voronoi diagram are randomly distributed, they may clump together, leaving uneven voids or forming undesirable patterns. Therefore, this study adopts the weighted Voronoi diagram based on the Centroidal Voronoi diagram, which ensures that the generators are well spaced. Fig. 11 shows the stippling results of different stippling methods. Fig. 11(a)–(c) were created using Bosch and Herman's algorithm, the ordered dithering algorithm, and the Floyd–Steinberg dithering algorithm, respectively. Fig. 11(d) was created using the weighted Voronoi diagram algorithm which is described in Fig. 12.

In the Centroidal Voronoi diagram, each generator lies exactly on the mass centroid of its Voronoi polygon. This centrality is an important characteristic of the Centroidal Voronoi diagram. The mass centroid $A_i$ of a Voronoi polygon $R_i$ with probability density $\rho(q_i)$ is defined as

$$A_i = \frac{\int_{R_i} q_i \rho(q_i) dR_i}{\int_{R_i} \rho(q_i) dR_i}, \tag{9}$$

where $q_i = \left(x^i, y^i\right)$ is a pixel in the Voronoi polygon $R_i$, i.e., $q_i \varepsilon R_i$, and $x^i$ and $y^i$ are the horizontal and vertical grid coordinates, respectively. The probability density $\rho(q_i)$ at pixel $q_i = \left(x^i, y^i\right)$ is calculated as

$$\rho(q_i) = \frac{w_R * R(q_i) + w_G * G(q_i) + w_B * B(q_i)}{255} \tag{10}$$

which converts the input RGB values of the corresponding pixels at a grayscale level. $w_R$, $w_G$, and $w_B$ are the weight parameters of the R, G, and B values, respectively, here set to $w_R = 0.21$, $w_G = 0.71$, and $w_B = 0.07$ as recommended in the ITU-BT.709-6 standard [25]. Each pixel has a grayscale level value between 0 (white) and 255 (black).

We observed that in the stippling stage, the darker regions of the image more naturally attract generators than brighter regions. To improve the stippling result, we adopted a new density function $\rho_{weight}(q_i)$ [19]. Given that $0 \le \rho(q_i) \le 1$ ranges from black to white, the new density function is defined as

$$\rho_{weight}(q_i) = 1 - \rho(q_i), \tag{11}$$

Note that the new probability density function $\rho_{weight}(q_i)$ assigns higher densities to black pixels rather than the white pixels. The new mass centroid $A_i$ is defined as

$$A_i^{weight} = \frac{\int_{R_i} q_i \rho_{weight}(q_i) dR_i}{\int_{R_i} \rho_{weight}(q_i) dR_i}. \tag{12}$$

Consequently, we modified Lloyd's iterative algorithm [26] as follows:

Fig. 13 displays non-weighted and weighted Voronoi diagrams generated by the proposed algorithms in the developed software. The number of generators was 500. Fig. 13(a) and (b) show the Voronoi diagrams with and without the overlaying original image, respectively. The corresponding weighted Voronoi

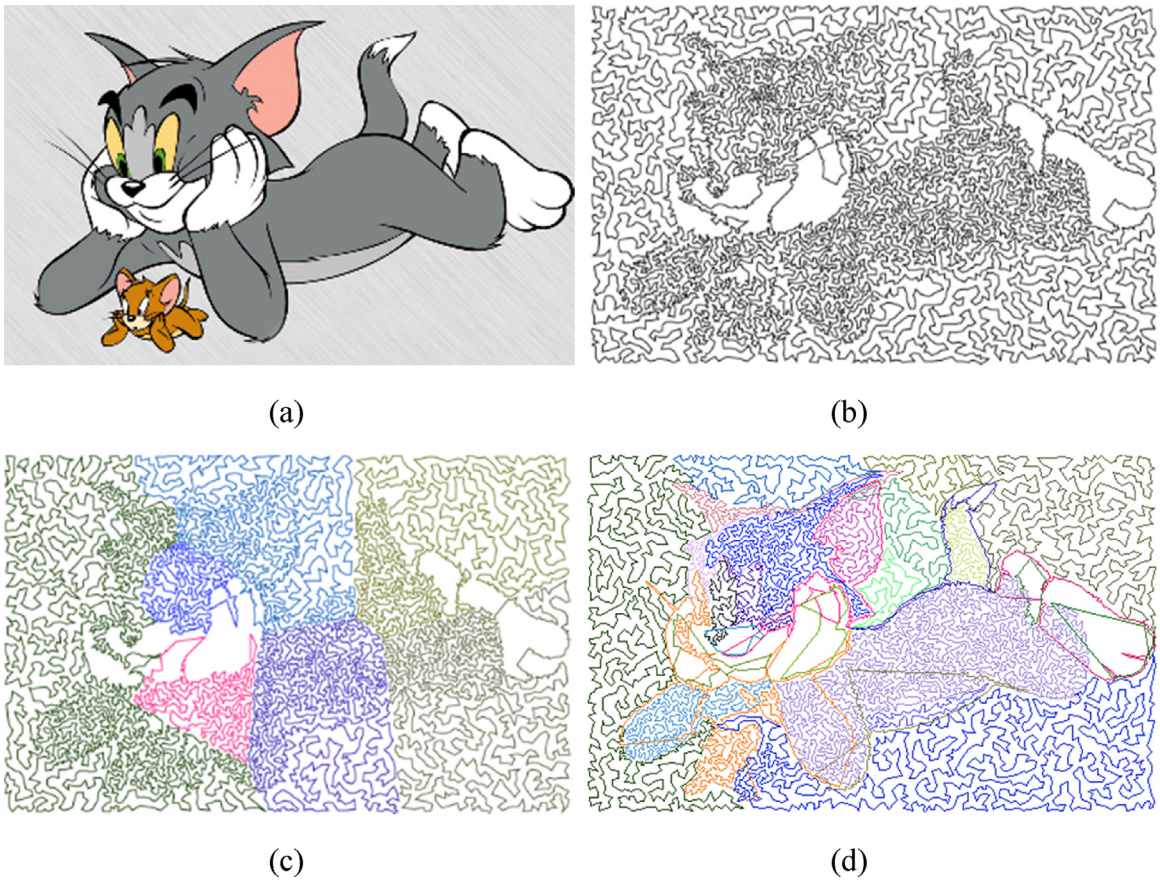


**Fig. 18.** Results of different algorithms: (a) input image, (b) a single CLD, (c) k-CLD by the segmentation using the k-means clustering algorithm, and (d) k-CLD by the proposed algorithm.

diagrams are presented in Fig. 13(c) and (d), respectively. It is clearly observed that the stippled dots in Fig. 13(d) are well spaced. The animation for the modified Lloyd's iterative algorithm shows how the generators and Voronoi polygons are updated at http://prof.pusan.ac.kr/user/optimus/research/modified-lloyd-iterative-algorithm.html [27].

### 3.3. Ant colony system for k-CLD

As mentioned above, the target image was partitioned into multiple regions. Subsequently, the target image was processed by the weighted Voronoi diagram and then each region was stippled. The stippled dots in each region were well spaced and must be connected to build a single CLD. As these dots are analogous to cities in TSP, an effective and efficient TSP can be used to construct the CLD for each region.

Considering that the algorithms can run on mobile phones in future and remain compatible with the rapid development of core technologies of modern microprocessors, we prioritized scalability and parallelism. Current implementation is developed on desktop computers with powerful multi-core microprocessors. In addition, various possible tradeoffs of the algorithm parameters (image quality, execution time of the application, and CPU usage) can be easily controlled even by users who are do not understand the complex algorithms explained here. In this study, TSP in each region of the images was solved by an ant colony system algorithm. There exist a good number of variants of ant colony optimization algorithms including ant system [28], min-max ant system [29], and rank-based ant system [28]. The ant colony system (ACS) algorithm is one of the well-known metaheuristic methods for TSPs [30]. The parameters in this algorithm can be controlled by users in various ways, and the independent ants are amenable to parallelism on various multi-core microprocessors in future research and applications. In our intensive experiments, ACS appears to find good solutions to variously shaped stippled drawings with a reasonable runtime.

Being a distributed algorithm, ACS can be easily applied to TSPs. In ACS, good TSP solutions are found by a set of cooperating agents called ants. Ants cooperate by depositing a pheromone at the edges of the TSP graph while building solutions. This pheromone provides an indirect communication channel for the cooperating ants [31].

Let $n$ denote the number of ants cooperating to draw a CLD. Initially, the pheromone is deposited over all possible paths connecting the stippled dots. Each ant then builds a feasible solution by choosing the next cities to visit, excluding the already-visited cities under pseudorandom proportional rules. The probability that ant $k$ visits dot $j$ from current dot $i$ can be expressed as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{{\tau_{ij}}^{\alpha}{\eta_{ij}}^{\beta}}{\sum_{c_{il}\in N_i^k}{\tau_{il}}^{\alpha}{\eta_{il}}^{\beta}} & \text{if } c_{il} \in N_i^k, \eta_{ij} = \dfrac{1}{d_{ij}}, \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where $d_{ij}$ is the Euclidean distance between dots $i$ and $j$, $N_i^k$ is the not-visited-yet neighborhood of ant $k$ at current dot $i$, and the parameters $\alpha$ and $\beta$ specify the relative importance of the pheromone vs. the inverse of distance.

$$j = \begin{cases} \arg\max_{l\in N_k}\left\{\tau_{il}\eta_{il}^{\beta}\right\}, & \text{if } q \le q_0 \\ J, & \text{otherwise} \end{cases}. \tag{14}$$

Here, $\tau_{il}$ is the pheromone and $q$ is a random variable uniformly distributed over [0,1] and $\eta_{il}$ is the inverse of the Euclidean distance

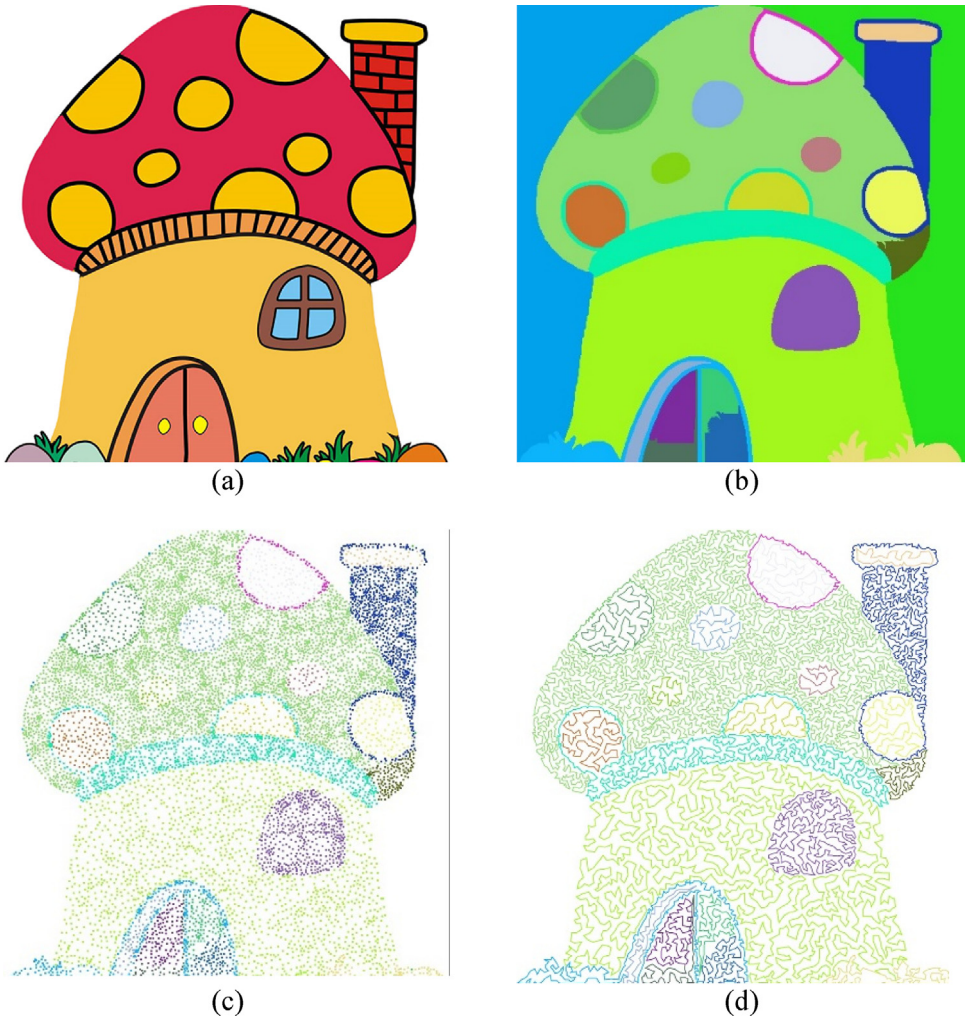


**Fig. 19.** (a) Input image, (b) segmentation result, (c) stippling result, and (d) k-CLD result.

between dots $i$ and $l$ and $q_0 \in [0, 1]$ is the threshold parameter. $J$ is a random variable selected according to the probability distribution $p_{ij}^k$, which favors edges that are shorter and have a higher level of pheromone trail:

When an ant traverses an edge, it updates the amount of pheromone on that edge according to the local-pheromone update rule:

$$\tau_{ij} = \left(1 - \xi\right)\tau_{ij} + \xi\tau_0, \tag{15}$$

where $\xi \in [0, 1]$ is a parameter and $\tau_0$ is the initial value of the pheromone.

During this iteration, all ants complete their tasks by producing their own feasible solutions. The second pheromone update is governed by offline-pheromone update rules.

$$\Delta\tau_{ij} = \begin{cases} (1-\varphi)\tau_{ij} + \varphi\Delta\tau_{ij} & \text{if } (i,j) \text{ belongs to the best tour} \\ 0 & \text{otherwise} \end{cases}, \tag{16}$$

where $\varphi \in (0, 1)$ is the pheromone decay coefficient. Here, $\Delta\tau_{ij} = 1/L_{best}$, where $L_{best}$ is the tour length of the best-performing ant.

The implemented ACS algorithm considers the pheromone strengths (equivalent to memory) and short edges (equivalent to visibility).

In this study, the number of cities varies by region. The desired number of ants can be decided from the quality of the output image and the capability and runtime of the computing system. Fig. 14 shows the difference in the image quality caused by the number of ants on the input images. Fig. 14(b) and (c) only show the part of the target image under conversion, which is represented as a rectangular area on the bottom of the target. Within 3 min of software execution, Fig. 14(b) and (c) are obtained employing 100 ants and 10 ants. The software tool has control over maintaining the number of ants and the execution time. The users should be able to compromise the image quality with the process parameters. For computational experiments in this study, the parameters are set as follows; $\alpha = 1, \beta = 2, q_0 = 0.9$, $\xi = 0.5$, $\varphi = 0.5$ and the total number of ant is 30.

## 4. Computational results

The proposed software is planned for implementation on various computing platforms including mobile phones, as mobile devices are gaining popularity and produce an enormous number of images. Therefore, such devices are a potentially strong market for artistic software. However, to demonstrate the effectiveness and utility of the proposed methodologies, we developed the current version on a desktop computer. Visual Studio 2013, and C++ has been used to implement the proposed on a desktop PC with an Intel(R) Core(TM) i7-4792 CPU@3.6 GHz and 16 GB memory and a Windows 10 Pro system. Fig. 15 is a screenshot of the developed software, which has an intuitive graphical user interface. Efforts

to improve the user interface and design for mobile devices are in progress.

To demonstrate the excellence of the proposed methodologies in their current implementation, Fig. 16 shows a computational comparison for two different algorithms to partition the stipplings. Fig. 16(a) and (b) show the stippled regions in the target image after segmentation by k-means clustering [32] and the k-CLD produced from these stipplings, respectively. Fig. 16(c) and (d) show the stippled regions in the target image after segmentation by the proposed methodologies and the k-CLD produced from these stipplings, respectively. For comparison purposes, the stipplings in Fig. 16(a) and (c) each have 27 clusters.

Fig. 17 visually compares the target image and images processed by the proposed methodologies. Figs. 17(a)–(d) display the input target image, segmentation result, stipplings in each segmented region, the final k-CLD result, respectively.

To illustrate the contributions of the proposed algorithm comparatively, the results of various algorithms are presented in Fig. 18. Given Fig. 18(a) as the input image, Fig. 18(b) is generated by the single CLD algorithm without the segmentation and stipplings using the weighted Voronoi diagram. In Fig. 18(c), k-means clustering algorithm has been employed for the segmentation, instead of the proposed segmentation algorithm. Fig. 18(d) is the result of the proposed algorithm including the segmentation, stippling, and TSP-solving. The assessment of the image quality can be subjective matters for the humans but it is demonstrated that the proposed algorithm can generate interesting results.

Another example is given in the following Fig. 19(a) shows the input image, which is segmented in Fig. 19(b). Then, it was stippled as shown in Fig. 19(c). Finally, the k-CLD has been given in Fig. 19(d), which shows the effectiveness of the proposed methodologies.

## 5. Conclusion

In this study, we have investigated whether mathematical and computational intelligence can be linked with artistic activities. To this end, we proposed a new artistic drawing technique, based on CLD, in which the artist can complete his sketch without lifting his pen from the paper. This technique is mainly applicable to black-and-white images. However, colored images from various mobile devices are gaining popularity, and mobile phone users might be interested in applying this new type of drawing, which is similar to CLD, to colored images captured by the cameras on their mobile devices.

The k-CLD methodologies proposed in this study are implemented on a desktop PC and are expected to solve three main challenges for obtaining high-quality k-CLD.

The k-CLD methodologies are proposed and described in this study. The first step is to condense the input image by segmenting it into multiple regions. Here $C_j \subset S$ is a segmented region and $S = (C_1, C_2 \cdots C_r)$. The segmentation is performed by an effective graph-based image segmentation algorithm. Each region is then processed by the weighted Voronoi diagram algorithm, which stipples the input image into small dots $P = (p_1, p_2 \cdots p_r)$, where $p_j$ is a set of stipplings in region $C_j$. An arbitrary region $p_j \subseteq P$ contains a set of stipplings, i.e., $p_j = (v_1, v_2 \cdots v_m)$, where each $v_i$ is a small dot belonging to $p_j$. In TSP, each $v_i$ is considered as a city. Economical tours among the cities of each region are obtained by the ACS algorithm. Finally, the small dots in the optimized tour of each region are connected to produce the k-CLD. The k-CLD is rendered in different colors for each region.

As a concluding comment, we wish to share our experience in conducting this study. Motivated by the possible link between the soft computing and the art, we conducted this research while considering the potential users and what is publicly popular. To avoid duplication of previous work, we conducted an extensive online search of existing studies. The problem was then divided into smaller problems, each of which was studied individually to obtain more effective and efficient solutions. Many developmental and computational efforts were invested in extending existing algorithms to solve the new problem.

## Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2015R1A2A1A10054253).

## References

[1] R. Bosch, A. Herman, Continuous line drawings via the traveling salesman problem, Oper. Res. Lett. 32 (2004) 302–303.
[2] C. Kaplan, R. Bosch, Tsp art, Bridg Conf. Proc. (2005) 303–310.
[3] B.E. Bayer, An optimum method for two-level rendition of continuous-tone picture'Int, Conf Commun. Conf. Rec. (1973) 11–26.
[4] R. Bosch, Connecting the dots: the ins and outs of TSP art, Proc. Bridg. (2008) 235–242.
[5] H. Li, D. Mould, Continuous line drawings and designs, Int. J. Creat. Interfaces Comput. Graph. 5 (2014) 16–39.
[6] J. Zhang, J. Hu, Image segmentation based on 2D otsu method with histogram analysis, 2008 Int. Conf. Comput. Sci. Softw. Eng. (2008) 105–108.
[7] V. Grau, A.U.J. Mewes, M. Alcaniz, R. Kikinis, S.K. Warfield, M. Alcañiz, Improved watershed transform for medical image segmentation using prior information, IEEE Trans. Med. Imaging 23 (2004) 447–458.
[8] J. Malik, S. Belongie, T. Leung, J. Shi, Contour and texture analysis for image segmentation, Int. J. Comput. Vis. 43 (2001) 7–27.
[9] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 888–905.
[10] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient graph-based image segmentation, Int. J. Comput. Vis. 59 (2004) 167–181.
[11] Z.T. Jastrzębski, Scientific Illustration A Guide For The Beginning Artist, Prentice Hall, 1985.
[12] V. Ostromoukhov, A simple and efficient error-diffusion algorithm, 28th Annu Conf. Comput. Graph. Interact. Tech. (2001) 567–572.
[13] V. Ostromoukhov, R.D. Hersch, Stochastic clustered-dot dithering, J. Electron. Imaging 8 (1999) 439–445.
[14] L. Velho, J. Gomes, Stochastic screening dithering with adaptive clustering, in: ACM SIGGRAPH 95, 1995, pp. 273–276.
[15] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (1982) (129–137.7).
[16] M. McCool, E. Fiume, Hierarchical Poisson disk sampling distributions, in: Conf Graph. Interface, 1992, pp. 94–105.
[17] M.P. Salisbury, M.T. Wong, J.F. Hughes, D.H. Salesin, Orientable textures for image-based pen-and-ink illustration, in: ACM SIGGRAPH 97, 1997, pp. 401–406.
[18] O. Deussen, S. Hiller, C. van Overveld, T. Strothotte, Floating points: a method for computing stipple drawings, in: Eurographics 2000, 2000, pp. 40–51.
[19] A. Secord, Weighted voronoi stippling, Second Int Symp. Non-Photorealistic Animat. Render (2002) 37.
[20] D. Kim, M. Son, Y. Lee, H. Kang, S. Lee, Feature-guided image stippling, Comput. Graph. Forum (2008) 1209–1216.
[21] C.H. Papadimitriou, The Euclidean travelling salesman problem is NP-complete, Theor. Comput. Sci. 4 (1977) 237–244.
[22] L. Xu, C. Lu, Y. Xu, J. Jia, Image smoothing via L0 gradient minimization, ACM Trans. Graph 30 (2011) 174:1–174:12.
[23] W.-M. Pang, Y. Qu, T.-T. Wong, D. Cohen-Or, P.-A. Heng, Structure-aware halftoning, ACM Trans. Graph 27 (2008) 89:1–89:8.
[24] R.W. Floyd, An adaptive algorithm for spatial gray-scale, Soc. Inf. Disp. (1976) 75–77.
[25] https://www.itu.int/rec/R-REC-BT.709-6-201506-I/en (Accessed December 14 2016).
[26] A. Okabe, B. Boots, K. Sugihara, S.N. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, John Wiley & Sons, 2009.
[27] http://prof.pusan.ac.kr/user/optimus/research/modified-lloyd-iterative-algorithm.html (Accessed August 20 2017).
[28] B. Bullnheimer, R. Hartl, C. Strauß, A new rank based version of the ant system – a computational study, Cent. Eur. J. Oper. Res. Econ. 7 (1997) 25–38.
[29] T. Stützle, H.H. Hoos, MAX–MIN ant system, Futur. Gener. Comp. Syst. 16 (2000) 889–914.
[30] M. Dorigo, L.M. Gambardella, Ant colonies for the travelling salesman problem, Biosystems 43 (1997) 73–81.
[31] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evolut. Comput. 1 (1997) 53–66.
[32] K. Krishna, M.N. Murty, Genetic k-means algorithm, IEEE trans. syst. man, Cybern. Part B Cybern. 29 (1999) 433–439.