Computers & Geosciences 105 (2017) 91–102

Contents lists available at ScienceDirect

Computers & Geosciences

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

# Fractal generator for efficient production of random planar patterns and symbols in digital mapping

Qiyu Chen[a,b,c], Gang Liu[a,c,*], Xiaogang Ma[d], Xinchuan Li[a,c], Zhenwen He[a,c]

[a] School of Computer Science, China University of Geosciences, Wuhan 430074, Hubei, China
[b] Institute of Earth Surface Dynamics, University of Lausanne, 1015 Lausanne, Switzerland
[c] Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, Hubei, China
[d] Department of Computer Science, University of Idaho, 875 Perimeter Drive MS 1010, Moscow, ID 83844-1010, USA

ARTICLE INFO

*Keywords:*
Digital cartography
Fractal generator
Random planar patterns
Iterated function systems
Similarity and randomness

ABSTRACT

In digital cartography, the automatic generation of random planar patterns and symbols is still an ongoing challenge. Those patterns and symbols of randomness have randomly variated configurations and boundaries, and their generating algorithms are constrained by the shape features, cartographic standards and many other conditions. The fractal geometry offers favorable solutions to simulate random boundaries and patterns. In the work presented in this paper, we used both fractal theory and random Iterated Function Systems (IFS) to develop a method for the automatic generation of random planar patterns and symbols. The marshland and the trough cross-bedding patterns were used as two case studies for the implementation of the method. We first analyzed the morphological characteristics of those two planar patterns. Then we designed algorithms and implementation schemes addressing the features of each pattern. Finally, we ran the algorithms to generate the patterns and symbols, and compared them with the requirements of a few digital cartographic standards. The method presented in this paper has already been deployed in a digital mapping system for practical uses. The flexibility of the method also allows it to be reused and/or adapted in various software platforms for digital mapping.

## 1. Introduction

The fractal geometry (Mandelbrot, 1967) is originated from the simulation of complex natural phenomena. It aims to explore the nondeterministic, nonlinear and self-similar phenomenon and process, and discover the inherent law or order. In the past 50 years, the fractal geometry theory and its derivative methods have been widely applied in various fields of geoscience (Carlson, 1991; Chen et al., 2016; Cheng et al., 2001; Cheng and Agterberg, 2009; Lovejoy and Schertzer, 2007; Subhakar and Chandrasekhar, 2016; Turcotte, 2005). The application of the fractal geometry has solved the relationship between whole and parts in a unique way, and has created many beautiful and novel images by using symmetry and self-similarity of spatial structure as well as the simulation models of various images, which the Euclidean geometry is hard to describe.

Maps are a primary form for representing fractal features of phenomena and processes in the natural world (Mandelbrot, 2004; Moriyama et al., 1997). The method of fractal geometry, since its origin, has achieved many breakthroughs and a wide range of applications in digital mapping and spatial data analysis (Batty, 1995; García-Morales, 2016; Kappraff, 1986; Longley and Mesev, 2002). Especially, using the fractal geometry to describe and analyze geographic boundaries, river systems, landforms, and other complicated mapping phenomena is a topic of broad interest (Gentil and Neveu, 2013; Jiang, 2015; Tarboton, 1996).

The Iterated Function Systems (IFS) is one of the important branches in fractal theory. Theoretically, it is believed that in the sense of affine transformation, the whole and parts of a geometric object have similar structures, and several affine transformations can be chosen to map the shape of the whole to the parts. Then, the shape of the whole can be simulated through the iteration combined by uncertainty and randomness (Chen et al., 2009; Martyn, 2003; Wang et al., 2015). The construction of Koch curve is a good example of IFS, which can produce a curve that has complex morphological characteristic through simple iterated processes (Fig. 1). In real world applications, IFS is commonly used to express coastlines, outlines of mountain, tree models and others (Darmanto et al., 2013; Siddiqi et al., 2014). Natural scenes tend to have random behaviors but without strict self-similarity. The IFS codes obtained by introducing random variables can be treated with random disturbances in a certain way to generate

* Correspondence to: School of Computer Science, China University of Geosciences, No.388, Lumo Road, Wuhan 430074, Hubei, China.
*E-mail address:* liugang67@163.com (G. Liu).

http://dx.doi.org/10.1016/j.cageo.2017.04.009
Received 30 August 2016; Received in revised form 26 February 2017; Accepted 23 April 2017
Available online 05 May 2017

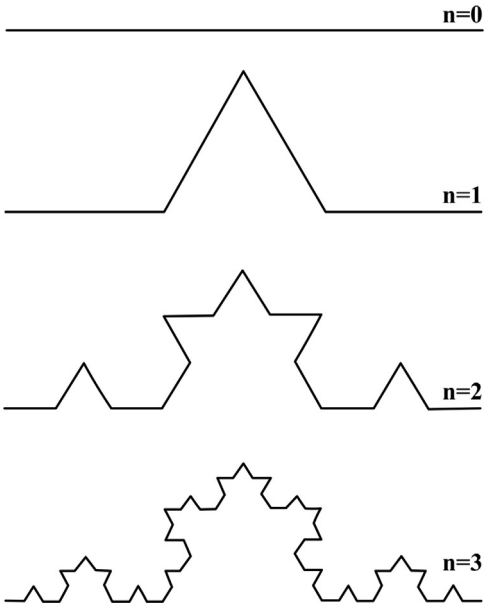


**Fig. 1.** Koch curve and its generator.

different natural scenes, which is a widely used method at present (Barnsley et al., 2005; Kya and Yang, 2001; Zhou et al., 2016).

## 2. Related works

Color covering and symbol filling are two major approaches to express different map elements in digital mapping using a GIS software (Kraak and Ormeling, 2010; Taylor, 2005). Symbols can offer abundant information, so they are more favored by cartographers and have been widely used in geological and geographical mappings. Many scholars have made in-depth study on the automatic drawing and filling of regular map patterns (Gustavsson et al., 2006; Hoelzel, 2004; Mihalynuk et al., 2006; Nass et al., 2011), and have set up controlled vocabularies and symbol libraries to meet the requirements of various domains of study (Głażewski et al., 2010; Li et al., 2009; Ma et al., 2010, 2012; Nayef and Breuel, 2013; Qiu et al., 2013). Hoelzel (2004) developed a simple but functional application, which automatically creates lithostratigraphic columns from field data used CorelDRAW graphic package. Mihalynuk et al. (2006) proposed a prototype set of geological symbols and the symbol set permitted easy and accurate production of standard geological maps in a GIS environment. Gustavsson et al. (2006) presented a comprehensive and flexible geomorphological combination legend that expands the possibilities of geomorphological mapping concepts. The symbol-based information could be digitally stored as a powerful database with thematic layers and attribute tables. Nass et al. (2011) focused on the simplification of mapping processes, and implemented the cartographic symbols drawing of planetary mapping data using GIS-based environments. Qiu et al. (2013) implemented the automation of geological drafting in CorelDRAW, which can visualize a large amount of data in a short period of time. To facilitate digital mapping, many professional software programs have been developed, such as AutoCAD, ArcGIS, MapInfo, CorelDRAW, and more. Those software programs are able to solve the drawing and filling problems of a large number of regular map patterns. However, the technologies for automatic generation and filling of random planar patterns and symbols are still not mature yet, and remain as a challenge for workers in the field of digital mapping. In order to further demonstrate the shortcomings of existing technologies, Fig. 2 is adopted and the corresponding explanations are as follow.

Fig. 2 shows several geological pattern symbols created in CorelDRAW and a geological section filled with those symbols. In Fig. 2, ① to ⑥ are simple regular pattern symbols, for which the technologies of automatic drawing and filling are well established by repetitive tiling of the same symbol unit. The strict regularity and symmetry of those symbols allow natural jointing of the titling units at the outboard edges, without irregular phenomena such as cracks, intersections and overlaps. ⑦ and ⑧ are simple random planar pattern symbols, which are constituted by mixed discrete symbols of sub-patterns. Though the patterns in ⑦ and ⑧ have some characteristics of randomness, by the arrangement of sub-patterns, the method of tiling is also applicable to fill those patterns in a map, such as the geological section in Fig. 2.

In contrast, ⑨ and ⑩ in Fig. 2 show two hand-painted random planar patterns and areas filled with them. The automatic drawing of those random patterns (such as the marshland in ⑨ and the cross-bedding in ⑩) are a more complicated task comparing with those in ① to ⑧. Moreover, the algorithms for generating those patterns are constrained by the shape features, cartographic standards and other conditions. Due to the limitation of automated methods, manual drawing is still widely used for this kind of patterns and symbols. In addition, to fill those patterns into a map, the repetitive tiling of the same symbol unit also cannot achieve satisfied result. As demonstrated in the area marked by red dashed line in the filling result of ⑨, the pattern is repetitively filled in the area, but the effect is too monotonous and does not meet the requirements for randomness. Similarly, there are obvious cracks in the area marked by red dashed line in the filling result of symbol ⑩, because the tiled symbol units cannot fit with each other on the edge due to its own asymmetry.

To address this challenge, the random characteristics of symbols ⑨ and ⑩ should be studied and reflected in the algorithms for the automatic drawing of them. This will also make it easy to obtain cartographic symbols that are ready for use in software programs. Furthermore, to achieve satisfied mapping result there should be separate approaches for the automatic filling of symbols ⑨ and ⑩ due to their own random characteristics. The method of unit symbol tiling can be used for the random symbol ⑨ that consists of discrete units, but it should be guaranteed that there are enough symbols of randomness to choose and the adjacent tiling units are not filled with the same symbol, thus ensuring the randomness of the overall filling effect to a large extent. For symbol ⑩, because it is continuous and asymmetric, the whole area to be filled should be taken as a single unit and an overall filling method should be adopted.

In this paper, the authors used both fractal theory and random IFS to propose a method for the automatic drawing of random planar patterns and symbols. In this method, a few random disturbance factors were added during the iterative process of IFS. The output images of IFS had a certain degree of randomness while also showed the characteristic of self-similarity. The marshland and trough cross-bedding patterns were taken as typical case studies in the implementation of the developed method.

## 3. Methodology

### 3.1. Self-affinity mapping and IFS

Assume that $T$: $R^n \rightarrow R^n$ is a linear transformation of $R^n$ (which can be expressed as a $n \times n$ matrix); $t$ is a vector of $R^n$, then a self-affinity mapping on $R^n$ can be defined as follow:

$$S(x) = T(x) + t \tag{1}$$

Given a limited compression mapping set $\{S_1, S_2, ...,S_m\}$, let $m \geq 2$, then it is called an IFS. If an IFS is constituted by a contractive affine transformation $\{S_1, S_2, ...,S_m\}$ of $R^n$, then the attractor $F$ is called a self-affine set. The construction process of self-affine curve $F$ is shown in Fig. 3. Affine function and IFS based on the self-affine set are of great help to the fractal interpolation, and have been effectively used to describe the outlines of mountains, coastlines and natural scenes (Gentil and Neveu, 2013; Zhou and Li, 2008).

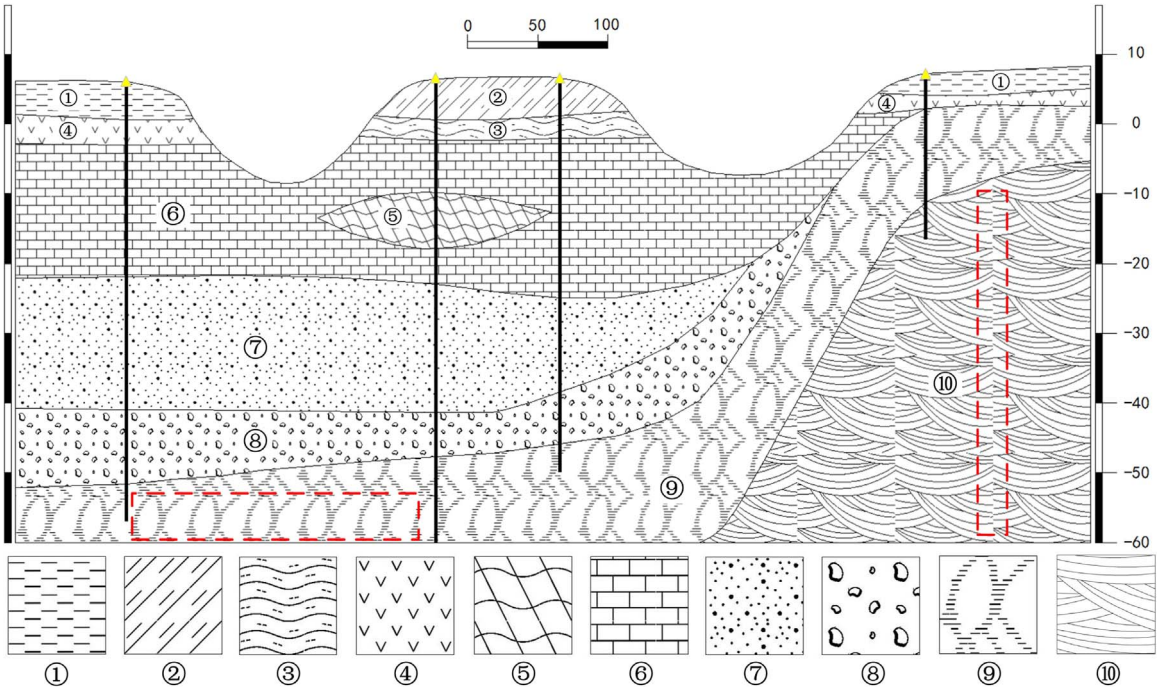


**Fig. 2.** Comparison of mapping results of different symbol types in CorelDRAW.

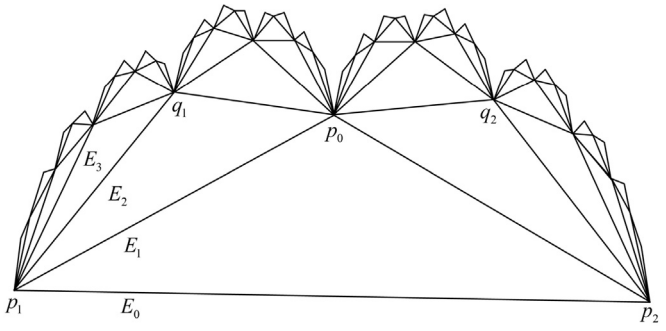


**Fig. 3.** Construction process of an affine curve.

In the IFS $\{X: S_1, S_2, \ldots, S_m\}$, linear contractive affine transformation is usually used to express compressed mapping, and the affine transformation $S_i: R^2 \to R^2$ on a 2D plane has the following form:

$$S_i\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}a_i & b_i\\ c_i & d_i\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + \begin{pmatrix}e_i\\f_i\end{pmatrix} = \begin{pmatrix}r_i\cos\theta_i & -s_i\sin\varphi_i\\ r_i\sin\theta_i & s_i\cos\varphi_i\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + \begin{pmatrix}e_i\\f_i\end{pmatrix} = A_iX + t_i, \quad i = 1, 2, \ldots, m, \tag{2}$$

where $a_i, b_i, c_i, d_i, e_i, f_i \in R^2$, $(x, y)$ is a point on a 2D plane; the linear transformation matrix $A_i$ represents rotation, proportion and reflection transformation; the vector $t_i$ represents translation transformation. A complex 2D fractal image that has self-similar structure can be generated by choosing a proper compressed mapping for repeated iterative calculations.

### 3.2. Fractal generator

The objects to be drawn in this paper are random planar patterns and symbols in geological and geographical mapping, which have no strict self-similarity. In order to realize characteristics of randomness in the output pattern, the paper introduces random disturbances control in the iterative process of IFS. Random noise interference is added in each iteration parameter to allow both the self-similar characteristic of affine transformation and randomness in the overall output pattern, thus increasing the feature of randomness in the result. However, the randomness must be controlled within a certain numerical range in order to let the fractal morphology of IFS evolve randomly in a reasonable range.

For the IFS with random disturbances, RIFS $\{X: S_1, S_2, \ldots, S_m | r_1, r_2, \ldots, r_m\}$, each compressed mapping is accompanied with a random disturbance $r_i$. As to the affine transformation of a 2D plane $S_i: R^2 \to R^2$ in the RIFS, the parameters $a_i, b_i, c_i, d_i, e_i, f_i$ in formula (2) must be obtained through a certain calculation with $r_i$, but the specific calculation form is decided by pattern features.

Hence, the fractal generator must include two parts: an IFS-based iterated function and a random number generator. The iterated function is abstracted from analyzing the features of random patterns and is used to express the self-similar characteristics of patterns. The affine transformation expressed by the iterated function can be one of the rotation, proportion, reflection and translation transformation or a combination of multiple transformations. A random number generator *random*() can generate random sequences that have a certain statistical characteristic. *random*() is applied to each iteration process of the IFS. It makes the iterative process random through the specific operation between the random numbers and the affine transformation parameters in the iterated function. Ultimately, the generated patterns are made to meet the requirements of both the self-similarity and the characteristics of randomness.

The construction of IFS, the selection of random number generator as well as the operation mode between various parameters and the random numbers during the iterative process are varied for different patterns of randomness, which will be decided by the features of the corresponding patterns.

We designed algorithms for the automatic drawing of two typical patterns of randomness, the marshland and the trough cross-bedding patterns in geological and geographical mappings, according to the thoughts and methods of the abovementioned fractal generator. In addition, the feasibility and effectiveness of the abovementioned method were tested through two case studies.

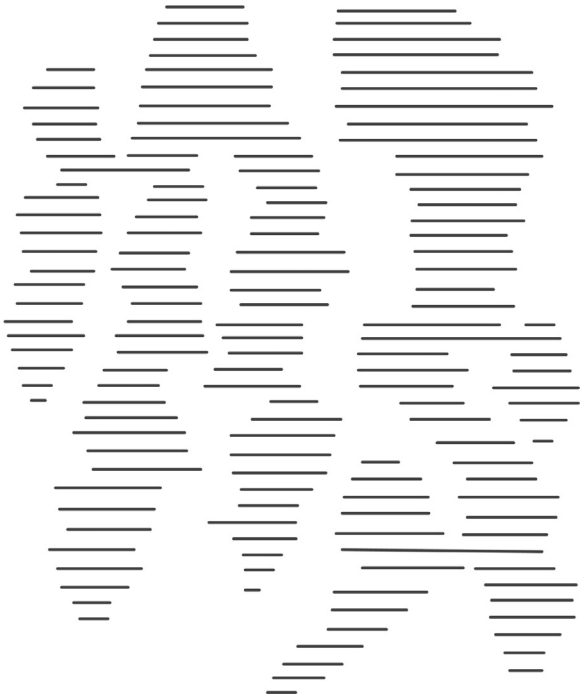

**Fig. 4.** Marshland patterns (According to GB/T, 14538-93, 1993).

### 3.3. Algorithms for generating marshland patterns

#### 3.3.1. Pattern characteristics

Fig. 4 shows the typical morphologies of marshland patterns in geological mapping, and the characteristic of which lies in that the boundary and internal filling are randomly varied, and these morphologies are the expression of the morphological distribution of marshland in reality. Though there are numerous morphological changes, their forms of expression are some random complex zones that are filled with horizontal lines, and the boundaries are not shown. Therefore, the key of drawing this pattern is how to get a series of lines that the coordinates $(X, Y)$ of starting points and the length of lines are randomly varied as well as are distributed at regular intervals in the longitudinal direction. Finally, these horizontal lines interlace around and generate a zone, which constitute a component for the overall marshland planar pattern. From Fig. 4, it can also be seen that the random varied range of the coordinates $(X, Y)$ of the starting points and the length of lines in each zone closely conforms to the Gaussian distribution.

#### 3.3.2. Design of algorithms

It is known from the above analysis that the drawing of the marshland patterns can be resolved into the drawing of several random zones filled by horizontal lines. Each zone consists of a number of horizontal lines which have the random coordinates $(X, Y)$ of starting points and randomly varied lengths. Multiple zones that consist of horizontal lines interlace around and then form the marshland patterns. Therefore, for the marshland patterns, a primary step is the automatic drawing of one zone that has this characteristic.

Fig. 5 offers the original state and the first three iterative processes of a zonal pattern. The original state is the starting coordinates $A(x_1, y_1)$ and $B(x_2, y_2)$ of the two horizontal lines with given lengths. One horizontal line will be inserted between two adjacent lines for every iterative step. Until the $n^{th}$ iteration process is finished, a zonal pattern is composed of the $2^n$ horizontal lines that have random lengths and randomly varied $X$-coordinates of the starting points.

It can be seen from the above processes that each newly generated line can be uniquely determined by its coordinate $(X, Y)$ of starting point and length $L$ of line. In which, the parameters with random characteristic are the $X$-coordinate of the starting point and length $L$. Assuming that *low* and *high* are the upper and lower indexes for an iteration, the iterated function of the fractal generator in drawing the marshland patterns can be defined as:

$$\begin{cases} x[mid] = (x[low] + x[high])/2 + \Delta p \\ y[mid] = (y[low] + y[high])/2 \\ l[mid] = random(l_\mu, l_\sigma) \end{cases}, \tag{3}$$

where, $\Delta p = random(x_\mu, x_\sigma)$ refers to the random offset of the $X$-coordinate; $l_\mu$ and $l_\sigma$ respectively express the mean and variance of length $L$; $x_\mu$ and $x_\sigma$ correspondingly show the mean and variance of $X$-coordinate of the line's starting point. $\Delta p$ and $l[mid]$ are both obtained by the random number generator $random(\mu, \sigma)$. $random(\mu, \sigma)$ can produce the random sequence that has the mean $\mu$ and variance $\sigma$ as well as conform to the Gaussian distribution.

Through the analysis above, the generating process of the random marshland pattern can be described as: the number $N$ of zones in horizontal direction, the number $n$ of lines in each zone, initial value $X_\mu$ (mean) of $X$-coordinate, degree of randomness $X_\sigma$ (variance) of $X$-coordinate, basic length $l_\mu$ (mean) of a line, degree of randomness $l_\sigma$ (variance) of length, fundamental distance $Xgap$ between two adjacent zones, and the initial points $p_{low}(x_{low}, y_{low})$ and $p_{high}(x_{high}, y_{high})$ are

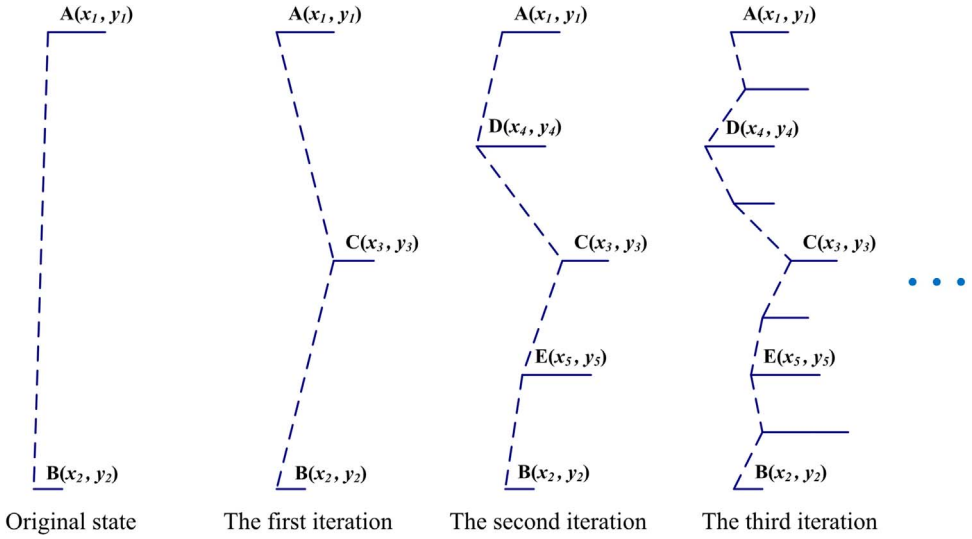


**Fig. 5.** Random iterative process of a zonal pattern.

given; starting from $p_{low}$ and $p_{high}$, a pattern of randomness with $N$ zones interlaced around is generated through controlling the random number generator $random(\mu, \sigma)$; each zone is composed by $n$ horizontal lines that have randomly variated $X$-coordinates of starting points and lengths of lines. Consequently, the core content of this algorithm includes two parts: (1) Fractal generator for one marshland zone (Algorithm 1), (2) An overall marshland pattern generator (Algorithm 2).

1. Fractal generator for one marshland zone (Algorithm 1). It is mainly composed by the recursive function FractalGeneratorMarsh($low$, $high$). The recursive entrance parameters $low$ and $high$ are the upper and lower indexes of the iteration, and are also the indexes of arrays. For example, when the number of lines needs to be draw $n = 17$, the initial $low = 0$ , $high = n - 1 = 16$; after one iteration, $mid = 8$, $x[8]$, $y[8]$, $l[8]$ can then be obtained, which means the horizontal line for the $index$ =8 is determined (lines 3–7). After one iteration, $mid$ is taken as the lower index of the first half and the upper index of the second half to carry out recursive calls, until $low \geq high$ ( lines 8–9). In which, the randomness of $X$-coordinate and length $l$ are adjusted by $random(\mu, \sigma)$. Because it is a binary recursive call, the number $n$ of lines must meet $n = 2^t$, and $t$ is the iteration times. The pattern on the right of Algorithm 1 is the drawing demonstration of a zone that is determined by $x[n]$, $y[n]$ and $l[n]$ after completing the recursion of the function FractalGeneratorMarsh($low$, $high$).
2. An overall marshland pattern generator (Algorithm 2). It can loop to draw a random marshland pattern circularly that is composed by $N$ zones interlaced around. Lines 3–6 in Algorithm 2 are the initialization of $x[0]$, $y[0]$, $l[0]$, $x[n-1]$, $y[n-1]$, $l[n-1]$ as well as $low$ and $high$ before the start of each loop. In which, the $X$-coordinates needs to increase $Xgap$ after each loop in order to ensure the distribution characteristics of the $N$ zones in the $X$-direction; the $Y$-coordinates of corresponding lines in the $N$ zones are consistent, which can ensure the lines interlaced are completed overlapped; length $L$ is obtained through the random number generator $random(l_\mu, l_\sigma)$, which can increase the randomness of the generated patterns. After all these, the recursive function FractalGeneratorMarsh($low$, $high$) is called to acquire the coordinate arrays $x[n]$ and $y[n]$ of starting points of $n$ horizontal lines as well as the length array $l[n]$ of corresponding lines in the $i^{th}$ zone (line 7). In the end, function DrawGraphic($x[n]$, $y[n]$, $l[n]$) is called to finish the visualization (line 8). Line 10 is a demonstration of a group of marshland patterns, which is composed by 8 zones interlaced around.

**Algorithm 1.** Fractal generator for marshland patterns

```
1:  function FractalGenerator_Marsh(low, high)
2:      if low < high then
3:          mid ← (low + high) / 2
4:          Δp ← random(X_μ, X_σ)
5:          x[mid] ← (x[low] + x[high]) / 2 + Δp
6:          y[mid] ← (y[low] + y[high]) / 2
7:          l[mid] ← random(l_μ, l_σ)
8:          FractalGenerator_Marsh(low, mid)
9:          FractalGenerator_Marsh(mid, high)
10:     end if
11: end function
```

**Algorithm 2.** The whole generating program of marshland patterns

**Input:** $N$: number of zones; $n$: number of lines in a zone; $X_\mu$: mean of $X$; $X_\sigma$: variance of $X$; $l_\mu$: mean of $L$ (line's length); $l_\sigma$: variance of $L$; $Xgap$: basic gap of 2 neighboring zones; $p_{low}(x_{low}, y_{low})$: lower point; $p_{high}(x_{high}, y_{high})$: upper point

**Output:** marshland patterns: $N$ zones and each zone consisted of $n$ lines with various lengths

```
1:  x[n] ← ∅; y[n] ← ∅; l[n] ← ∅
2:    for i: =0 → N such that i < N do
3:        x[0] ← x_low + i*Xgap; x[n − 1] ← x_high + i*Xgap
4:        y[0] ← y_low; y[n − 1] ← y_high
5:        l[0] ← random(l_μ, l_σ); l[n − 1] ← random(l_μ, l_σ)
6:        low ← 0; high ← n − 1
7:        FractalGeneratorMarsh(low, high)
8:        DrawGraphic(x[n], y[n], l[n])
9:    end for
10:
```

### 3.4. Algorithms for generating trough cross-bedding patterns

#### 3.4.1. Pattern characteristics

Cross-bedding is also called inclined bedding. It is composed by a series of laminas that are skew with each other at the boundaries of bed series, and the inclined bed series can be combined by mutual overlap, interlacement, and incision. The cross bedding can be divided into planar cross-bedding, wedge shaped cross-bedding, trough cross-bedding, etc. according to the forms and characters of the bed series and the upper and lower divisional plane. The trough cross-bedding is the

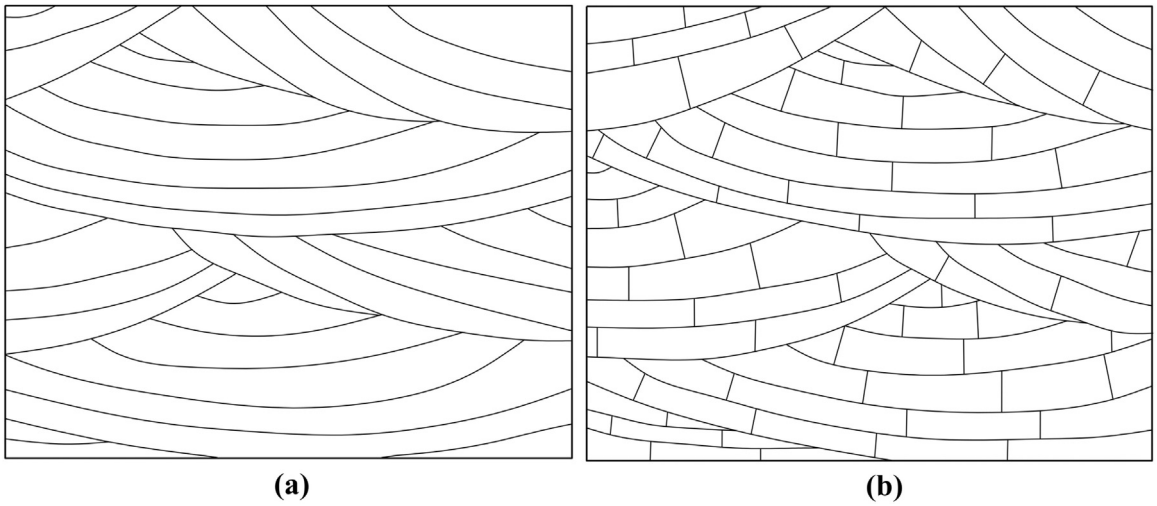


**Fig. 6.** Trough cross-bedding patterns: (a) shows the typical trough cross-bedding pattern; (b) shows the geological pattern of limestone trough cross-bedding following the geologic map symbolization in Federal Geographic Data Committee (2006).

most complex type among all those types. Its features are: the thickness of a single bed series changes quickly, and the lower boundaries of each bed series are concave downward; therefore, it has a distinct trough eroded base. The laminated beddings of the bed series can also both parallel and intersect the undersurface of the bed series with changeable inclined directions, and have obvious randomness.

Fig. 6(a) shows the typical trough cross-bedding pattern in geological thematic map. It can be seen from the figure that its beddings is formed by the interlacement of the middle, left-leaning and right-leaning bed series. The laminated beddings within the three parts are roughly paralleled, but have different inclined angles which are random. In addition to inclined angles, the gaps between those laminated beddings are also distinguishing and the change of gaps is also random. The characteristics of similarity and randomness exactly conform to the features of problems that fractal generator is able to solve, as described in the Section 3.2. After filling other symbols inside the trough cross-bedding, different geological patterns that are able to express more meanings can be achieved (Fig. 6(b)). However, the key of the study lies in the automatic drawing of bedding's frameworks.

### 3.4.2. Design of algorithms

According to the above analysis of the trough cross-bedding patterns, the similarity of this type of patterns is embodied in that each of the laminated beddings within the middle, left-leaning and right-leaning bed series is composed of approximately paralleled arcs. Its randomness is reflected in that the inclined angle of each arc within the laminated beddings differs, and the distance between two laminated beddings also varies. For the bedding patterns of each bed series, the process of drawing can be regarded as generating an arc $\widehat{AB}$ by taking a point $P$ as the center of a circle with a radius $R$, and filling the drawing area $F$ with the arc $\widehat{AB}$. The coordinate of each center $P$ should be disturbed randomly by $random(\mu, \sigma)$ to realize random inclined angles among laminated beddings. The descending distance $d$ of radius $R$ each time should vary randomly within a certain range to make random changes of the gaps between laminated beddings. We choose three center points $p_M$, $p_L$, $p_R$ from the middle, left and right, and loop drawing is carried out in order, and then we can acquire the trough cross-bedding that covers the entire filling area $F$.

With those parameters, the automatic drawing algorithm of the trough cross-bedding patterns mainly includes: initializing the center points, fractal generator, and the topology updating of $F$.

1. Initializing the center points (Fig. 7). We first confirm the bounding rectangle of the filling area $F$, and assume the line that has the same direction as the bending direction of the bed series is the center axis. Then we choose a point $p_M$ on the radial on the positive direction of the center axis as a middle center point, and its coordinate is $(x_M, y_M)$. On the line that go through the $p_M$ and is perpendicular to the center axis, we choose the point $p_L$ on the left line of the center axis and the point $p_R$ on the right as the left and right center points respectively, and their coordinates are $(x_L, y_L)$ and $(x_R, y_R)$ correspondingly. Make circumcircles of $F$ with $p_M$, $p_L$, $p_R$ as the centers of the circle respectively, and we can confirm the radiuses of the three circumcircles: $R_M$, $R_L$, $R_R$ which will respectively correspond to the initial radiuses for drawing the beddings with $p_M$, $p_L$, $p_R$ as the center points. It should be especially noted that the distance between the initial points $p_M$, $p_L$, $p_R$ and the filling area $F$ will affect the bending angle of the bedding: the larger the distance, the gentler the bedding, and vice versa. In addition, the distances between $p_M$ and $p_L$ as well as $p_M$ and $p_R$ decide the inclined degree of the left and right bed series: the larger the distance, the greater the inclined degree.
2. The fractal generator for trough cross-beddings (Algorithm 3). When drawing each laminated bedding in relevant bed series with $p_M$, $p_L$, and $p_R$ as the center points, the function

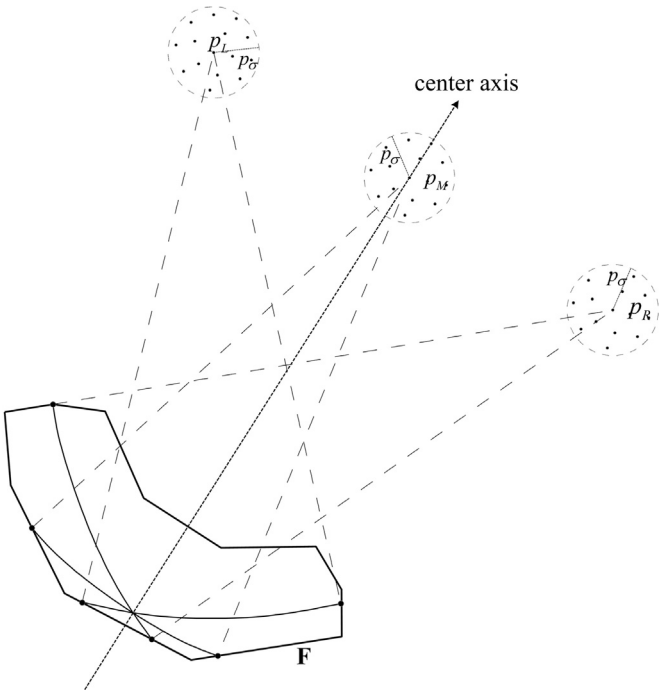


**Fig. 7.** Initialization of the central drawing points and their random iterative processing.

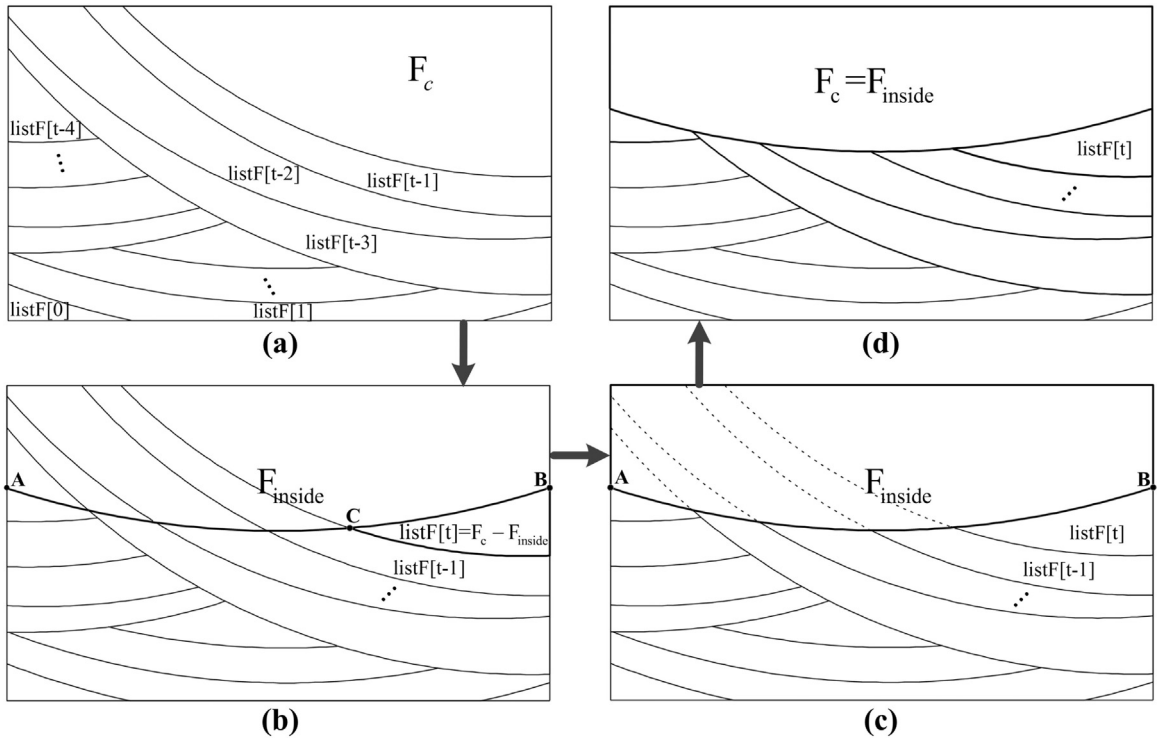


**Fig. 8.** Filling and upgrading process of the filling area $F$.

FractalGeneratorCross−bedding$(p, p_\sigma, R, d, d_\sigma)$ is used to determine each central drawing point $p'(x', y')$ and radius $R'$. On the basis of the last central drawing point $p(x, y)$, new central drawing point $p'(x', y')$ will appear in the circular region with $p$ as the center and $p_\sigma$ as the radius (shown in Fig. 7), through the random disturbance of the random number generator $random(p, p_\sigma)$. $p_\sigma$ will decide the random variation degree of new coordinates, lines 3–4 offer the solving process of $p'(x', y')$. In order to realize the random variation of the gaps between laminated beddings, the decreasing distance of the drawing radius $R$ each time should be varied randomly within a certain range. Adding random disturbance to the basic decreasing distance $d$, new drawing radius $R'$ can be expressed in the formula shown in lines 5–6. Algorithm 3 realizes the random variations of inclination and gaps of laminated beddings, thus allowing similarity and randomness among beddings.

**Algorithm 3.** Fractal generator for trough cross-beddings

**Input:** $p(x, y)$**:** last center point; $p_\sigma$: variance of $p$; $R$: last radius; $d$: basic decreasing step of $R$; $d_\sigma$: variance of $d$
**Output:** $p'(x', y')$: new center point; $R'$: new radius
**1: Function:** FractalGeneratorCross−bedding$(p, p_\sigma, R, d, d_\sigma)$
**2:** **if** continue drawing **then**
**3:** $x' \leftarrow Random(x, p_\sigma)$
**4:** $y' \leftarrow Random(y, p_\sigma)$
**5:** $d' \leftarrow Random(d, d_\sigma)$
**6:** $R' \leftarrow R - d'$
**7:** **end if**
**8:** **return** $(p', R')$
**9: end function**

3. Topology updating of $F$ (Algorithm 4). The drawing process of a laminated bedding is the topology computing and upgrading process between the arc $\overset{\frown}{AB}$ and the filling area $F$, filled area list $listF$, current filling area $F_c$. We make a circle with $p'(x', y')$ as the center, $R'$ as the radius, and get the intersecting arc $\overset{\frown}{AB}$ with the filling area $F$. Fig. 8(a) shows the initiative status before the $t^{th}$ drawing, and $listF$ stores the list of areas drawn during the previous $t-1$ times; $F_c$ is the remaining part after splitting $F$ in the previous $t-1$ times. The function FillingUpdating$F(F, \overset{\frown}{AB}, listF, F_c)$ offers the specific process of drawing laminated beddings. As shown in Fig. 8(b), we use $\overset{\frown}{AB}$ to separate $F$ into two halves, and get the inside area $F_{inside}$ of the $\overset{\frown}{AB}$; using the $F_{inside}$ to split the current filling area $F_c$, we can obtain the $t^{th}$ filling area object $listF[t] = F_c - F_{inside}$ (lines 2–5). Since that $\overset{\frown}{AB}$ might still be intersected with the previous $t-1$ area objects, and then previous $t-1$ area objects in the $listF$ should be judged and updated (Fig. 8(c)). Lines 6–11 describe the whole process of $listF$ updating. Ultimately, we update the current filling area to be $F_c = F_{inside}$, and the drawing is finished (Fig. 8(d)).

**Algorithm 4.** $F$ filling and topology updating

**Input:** $F$**:** area to be filled; $\overset{\frown}{AB}$: intersecting arc; $listF$: list of filled area; $F_c$: current filling area
**Input:** updated $listF$ and $F_c$
**1: Function:** FillingUpdating$F(F, \overset{\frown}{AB}, listF, F_c)$
**2:** $F_{inside} \leftarrow$ the inside polygon of $\overset{\frown}{AB}$ in $F$
**3: if** there are intersecting points between $F_c$ and $\overset{\frown}{AB}$ **then**
**4:** $F' \leftarrow F_c - F_{inside}$
**5:** $listF.add(F')$
**6:** $n \leftarrow$ the count of $listF$
**7:** **for** $t$: $=0 \rightarrow n-1$ such that $t \neq n-1$ **do**
**8:** **if** there are intersecting points between $listF[t]$ and $\overset{\frown}{AB}$ **then**
**9:** $listF[t] \leftarrow listF[t] - F_{inside}$
**10:** **end if**
**11:** **end for**
**12: end if**
**13:** $F_c \leftarrow F_{inside}$
**14:** **return** $(listF, F_c)$
**15: end function**

Algorithm 5 describes the complete filling process of the filling area $F$. We start drawing from the middle center point $p_M$, and loop $N_M$ times. $N_M$ is the number of the laminated beddings in one bed series. And then we make the central drawing points variates randomly each time through using the function FractalGenerator_Crossbedding $(p_M, p_\sigma, R_M, d, d_\sigma)$ , and also make the decreasing distance of the drawing radius random. ObtainArcAB$(F, p_M, R_M)$ is used to obtain the intersecting arc $\widehat{AB}$ between the circle which is with $p_M$ as the center and $R_M$ as the radius and the $F$. If the $\widehat{AB}$ exists, the FillingUpdating$F(F, \widehat{AB}, listF, F_c)$ will be called for drawing; if $\widehat{AB}$ does not exist, the drawing will end at the middle center point, thus let the flag variable $flag_M = 0$ and break out the loop of the $p_M$(lines 6–14). Same treatment will be carried out to the left and right center points successively (line 15–32). Draw circularly based on the order from

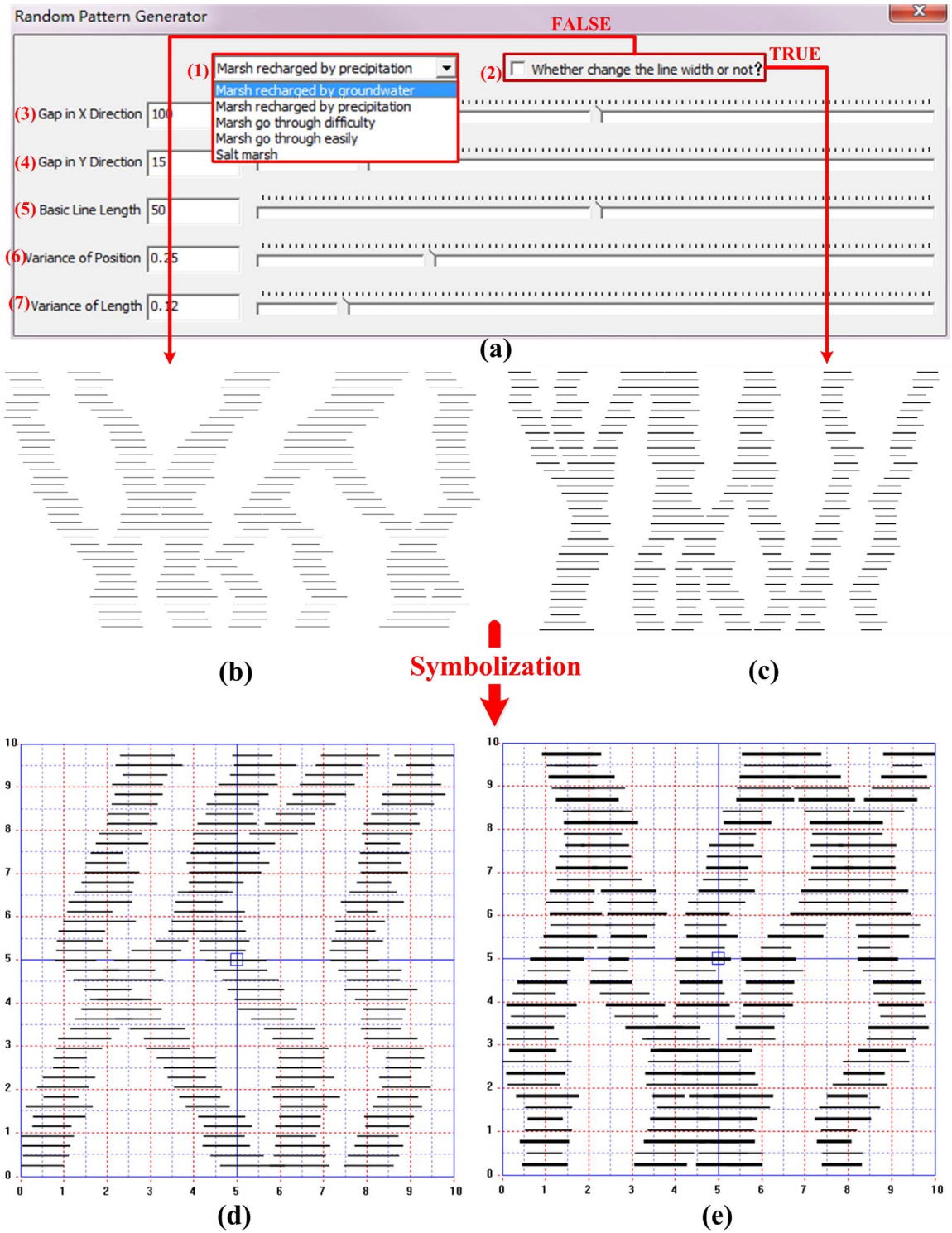


**Fig. 9.** User interface and pattern symbolization. (a) shows parameters on the user interface: (1) five different types of marshland patterns given in a national standard (GB/T, 14538-93, 1993); (2) whether change the line width or not; (3) basic gap between each zone in $X$ direction; (4) gap between lengthways lines; (5) mean value of line's length; (6) variance of the coordinates of lines; (7) variance of the changes of line's length. (b) and (c) show the different resulting outputs when the button (2) is switched on or off. (d) and (e) are the random marshland patterns after being symbolized.

middle, left and right until $flag_M == 0 \&\& flag_L == 0 \&\& flag_R == 0$, which means there is no intersecting points between $F$ and the circles with $p_M$, $p_L$, $p_R$ as the centers and $R_M$, $R_L$, $R_R$ as the radius respectively. Then the condition of line 5 is not met, and the loop will be ended and the drawing is finished. Ultimately, we carry out visualization of all area objects in the *listF*, and then we can obtain the trough cross-bedding patterns that is needed.

**Algorithm 5.** The whole generating program of trough cross-bedding patterns

**Input:** $F$**:** area to be filled
**Input:** *listF***:** list of filled areas
**1:** obtain the center axis of $F$
**2:** calculate the initial coordinates of 3 center points: $p_M(x_M, y_M)$, $p_L(x_L, y_L)$, $p_R(x_R, y_R)$; ascertain the corresponding radiuses: $R_M$, $R_L$, $R_R$
**3:** $flag_M = flag_L = flag_R = 1$
**4:** $F_c = F$, $listF \leftarrow \varnothing$
**5:** **while** $flag_M == 1 || flag_L == 1 || flag_R == 1$ **do**
**6:** **for** $i$: $=0 \rightarrow N_M$ **do**
**7:** $(p_M, R_M) \leftarrow$ FractalGenerator_Crossbedding$(p_M, p_\sigma, R_M, d, d_\sigma)$
**8:** $\widehat{AB} \leftarrow$ ObtainArcAB$(F, p_M, R_M)$
**9:** **if** $\widehat{AB} \neq NULL$ **then**
**10:** $(ListF, F_c) \leftarrow$ Filling_Updating$F(F, \widehat{AB}, listF, F_c)$
**11:** **else**
**12:** $flag_M = 0$, **break**
**13:** **end if**
**14:** **end for**
**15:** **for** $i$: $=0 \rightarrow N_L$ **do**
**16:** $(p_L, R_L) \leftarrow$ FractalGenerator_Crossbedding$(p_L, p_\sigma, R_L, d, d_\sigma)$
**17:** $\widehat{AB} \leftarrow$ ObtainArcAB$(F, p_L, R_L)$
**18:** **if** $\widehat{AB} \neq NULL$ **then**
**19:** $(ListF, F_c) \leftarrow$ Filling_Updating$F(F, \widehat{AB}, listF, F_c)$
**20:** **else**
**21:** $flag_L = 0$, **break**
**22:** **end if**
**23:** **end for**
**24:** **for** $i$: $=0 \rightarrow N_R$ **do**
**25:** $(p_R, R_R) \leftarrow$ FractalGenerator_Crossbedding$(p_R, p_\sigma, R_R, d, d_\sigma)$
**26:** $\widehat{AB} \leftarrow$ ObtainArcAB$(F, p_R, R_R)$
**27:** **if** $\widehat{AB} \neq NULL$ **then**
**28:** $(ListF, F_c) \leftarrow$ Filling_Updating$F(F, \widehat{AB}, listF, F_c)$
**29:** **else**
**30:** $flag_R = 0$, **break**
**31:** **end if**
**32:** **end for**
**33:** **end while**

## 4. Implementation and application analysis

The workflows and algorithms for drawing the two abovementioned patterns were realized in C++ language and implemented in QuantyView, a software platform for editing geological thematic maps.

### 4.1. Marshland patterns

Fig. 9(a) shows the user interface for generating the marshland patterns. A user can flexibly set up the input parameters by using the slide bars, and offer these parameters to the random marshland pattern generator described in Section 3.3. Following each change in the parameters, the drawing function will be called to generate a new morphology in the resulting pattern.

Fig. 9(b) and (c) show the changes in result when the button "(2) Change the line width of patterns" is switched on and off respectively. More varieties of marshland patterns can be obtained through combining this operation with the five types of marshlands in the dropdown list in (1) (also see Fig. 10 for details of those types). Then the forms of patterns generated will be further extended.

In order to make the generated patterns easy to use in the processes of digital mapping, the patterns generated need to be symbolized, and stored in a symbol library. We chose the generated patterns that meet the cartographic standards by controlling the interface parameters, and selected several forms can be for each type of patterns in order to meet practical needs. Fig. 9(d) and (e) are the random marshland patterns after being symbolized.

Fig. 10 shows a comparison between the marshland symbols of a national standard (GB/T, 14538-93, 1993) and the pattern that is generated by this method. We draw the random marshland patterns according to the five types given in the dropdown list (1) in Fig. 9, and for each type in that list the drawing results of three different patterns are given in Fig. 10. The three patterns differ from one another, and have the obvious characteristics of randomness. The comparison shows that the marshland patterns generated in this work meet to the requirements of the national standard, and have the flexibility to address specific needs of digital mapping in practice. Filling marshland symbols into a map layer is a different task compared to the generation of those symbols. Fig. 11 shows a thematic map of marshland that is filled with several forms of random marshland symbols.

### 4.2. Trough cross-bedding patterns

Fig. 12(a) shows the user interface of generating the trough cross-bedding patterns. We can flexibly control the parameters through the slide bars, and offer these parameters when drawing the random trough cross-bedding patterns as mentioned in Section 3.4. Operations on each slide bar will refresh the drawing function, and the pattern will be re-drawn. Even with the same parameter setting, the result obtained by the random number generator is different each time due to the random function in the algorithm, and the resulting patterns will be different each time as well.

Fig. 12(b) shows the drawing result of trough cross-bedding pattern with arbitrary boundary corresponding to relevant input parameters in the dialog. We can see that the bedding width is random and the inclined directions of the laminated beddings within one bed series are also random. However, the randomness is within a controllable range through the parameters.

The trough cross-beddings patterns can be overlapped with additional rock type symbols on a geological section. Fig. 13 depicts filling results with such overlapped patterns and symbols. Those results show that the work is capable to meet the specific requirements for mapping trough cross-beddings in cartographic standards, such as those listed in the Appendix A of Federal Geographic Data Committee (2006).

### 4.3. Discussion

As mentioned above, the mapping results of the two case studies meet the national and community cartographic standards well. In terms of mapping methods, our method overcomes the shortcomings of previous research on automatic drawing and filling of random map patterns (as illustrated in Fig. 2).

The first challenge comes from the automatic production of random planar patters and symbols in digital geological and geographical mappings. The basic idea is derived from fractal theory, and the fractal generator for efficient production of random planar patterns and symbols was designed by adding random disturbance factors in the IFS. More complicated patterns were generated due to interlacing different patterns

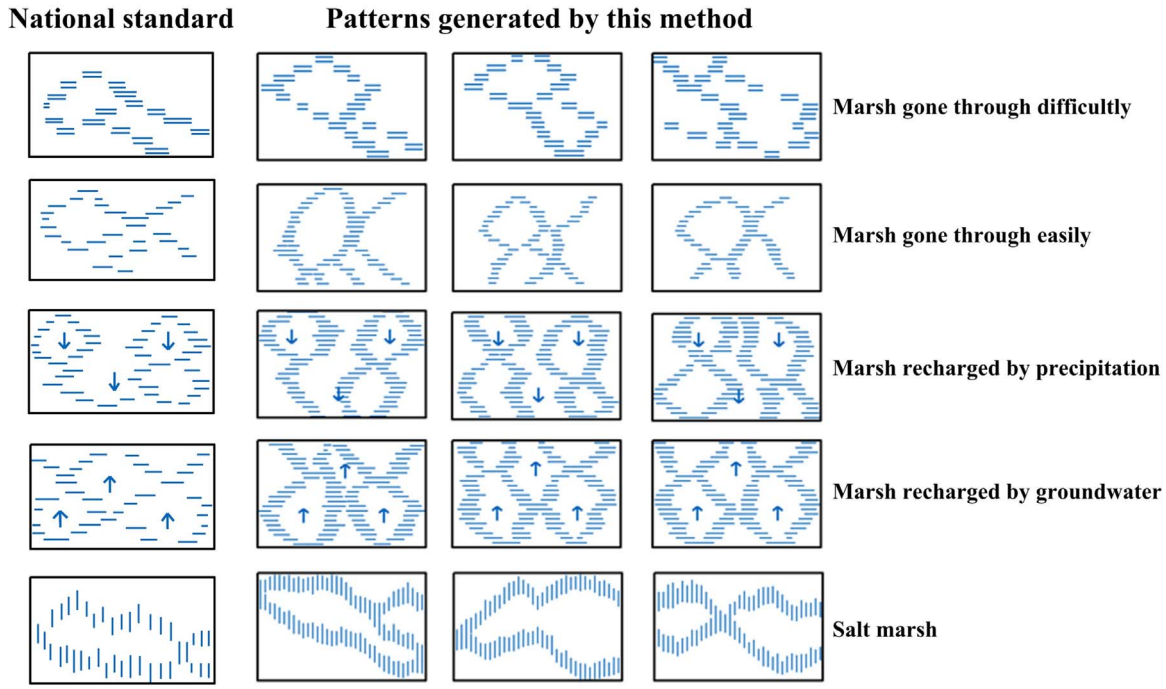


**Fig. 10.** Comparison between the marshland patterns generated in this work and the patterns in a national standard (GB/T, 14538-93, 1993).

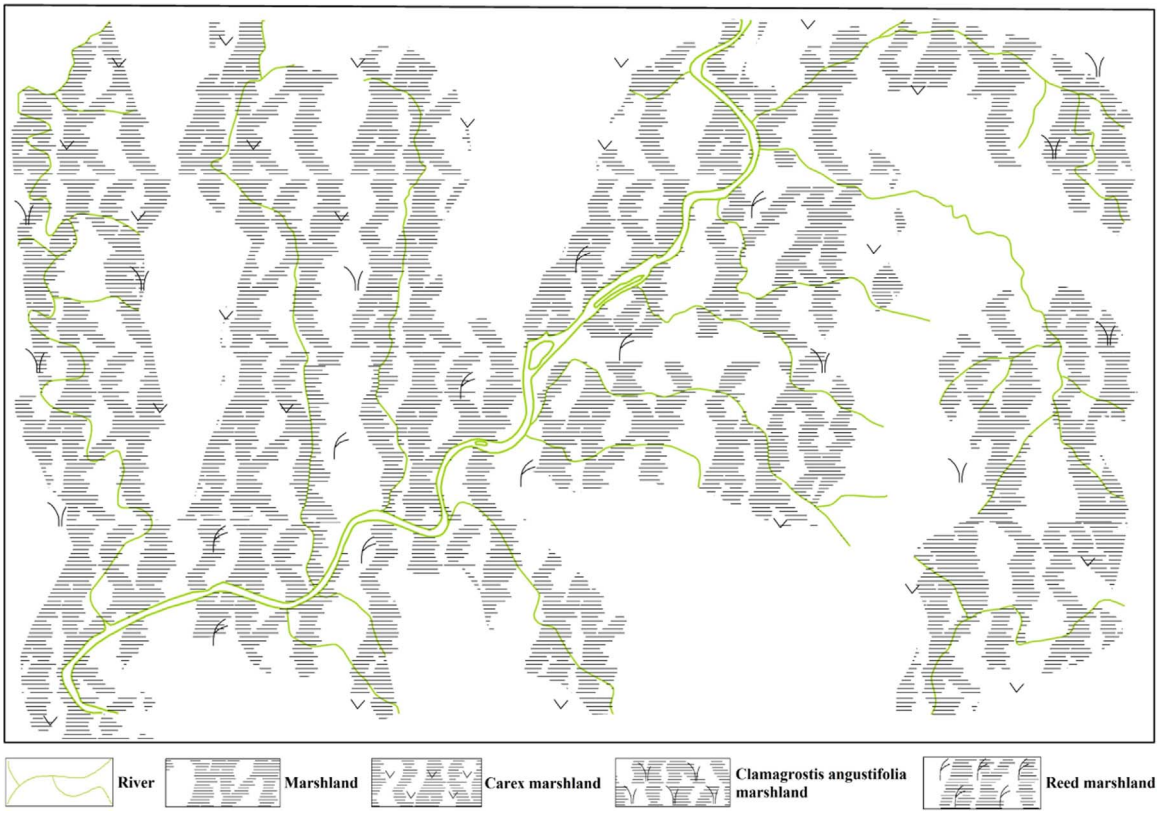


**Fig. 11.** Practical applications - thematic map of marshlands.

and cutting with boundaries. The friendly user interfaces were designed to control the randomness and the other parameters. Random patterns are encouraged in this work, but their randomness must be limited within a certain range. The abundant symbols of randomness were efficiently produced which conform to the national and community cartographic standards (As shown in Figs. 10 and 12). These symbol sets laid a foundation for the following maps drawing.

Filling these symbols of randomness into a map is the second challenge. As demonstrated in the area marked by red dashed line in the filling result of 9 and 10 in Fig. 2, both effects cannot meet the requirements of cartographic standards. Comparing Fig. 11 with the filling result of 9 in Fig. 2, our method overcome this challenge satisfactorily due to the enough symbols of randomness produced from the abovementioned method, thus ensuring the randomness of the overall

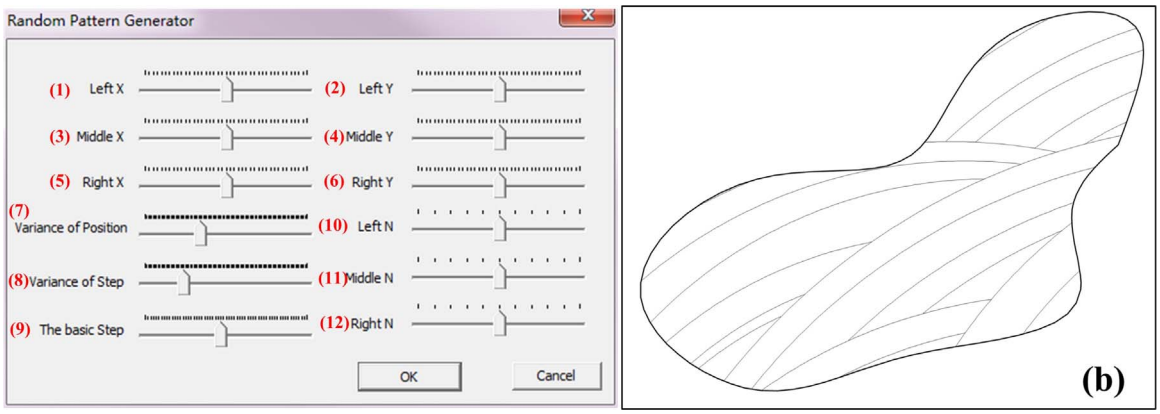


**Fig. 12.** User interface for generating trough cross-bedding patterns. (a) shows parameters on the user interface: (1)-(6) basic coordinates $p_M$, $p_L$, and $p_R$ of middle, left and right center points; (7) randomness $p_\sigma$ of coordinates of the three center points; (8) randomness $d_\sigma$ of the decreasing distance of the drawing radius; (9) mean decreasing distance $d$ of the drawing radius; (10)-(12) numbers $N_M$, $N_L$, and $N_R$ of the laminated beddings of the middle, left and right bed series. (b) shows the drawing result **of the t**rough cross-bedding pattern with arbitrary boundary according to the relevant parameters inputted in the dialog.

filling effect to a large extent. The method of unit symbol tiling was used for the random symbol 9 since it consists of discrete units. However, the tiled symbol units cannot fit the symbol 10 due to its continuity and asymmetry. An overall filling method was adopted and the challenge from symbol 10 illustrated in Fig. 2 was conquered (see Fig. 13).

The two typical case studies were developed to demonstrate the effectiveness and practicability of the method presented in this paper nicely. This method is also suitable for the other random planar patterns and symbols with the similar characteristics. These patterns focused on in this work are a part of the whole fractal patterns. A part

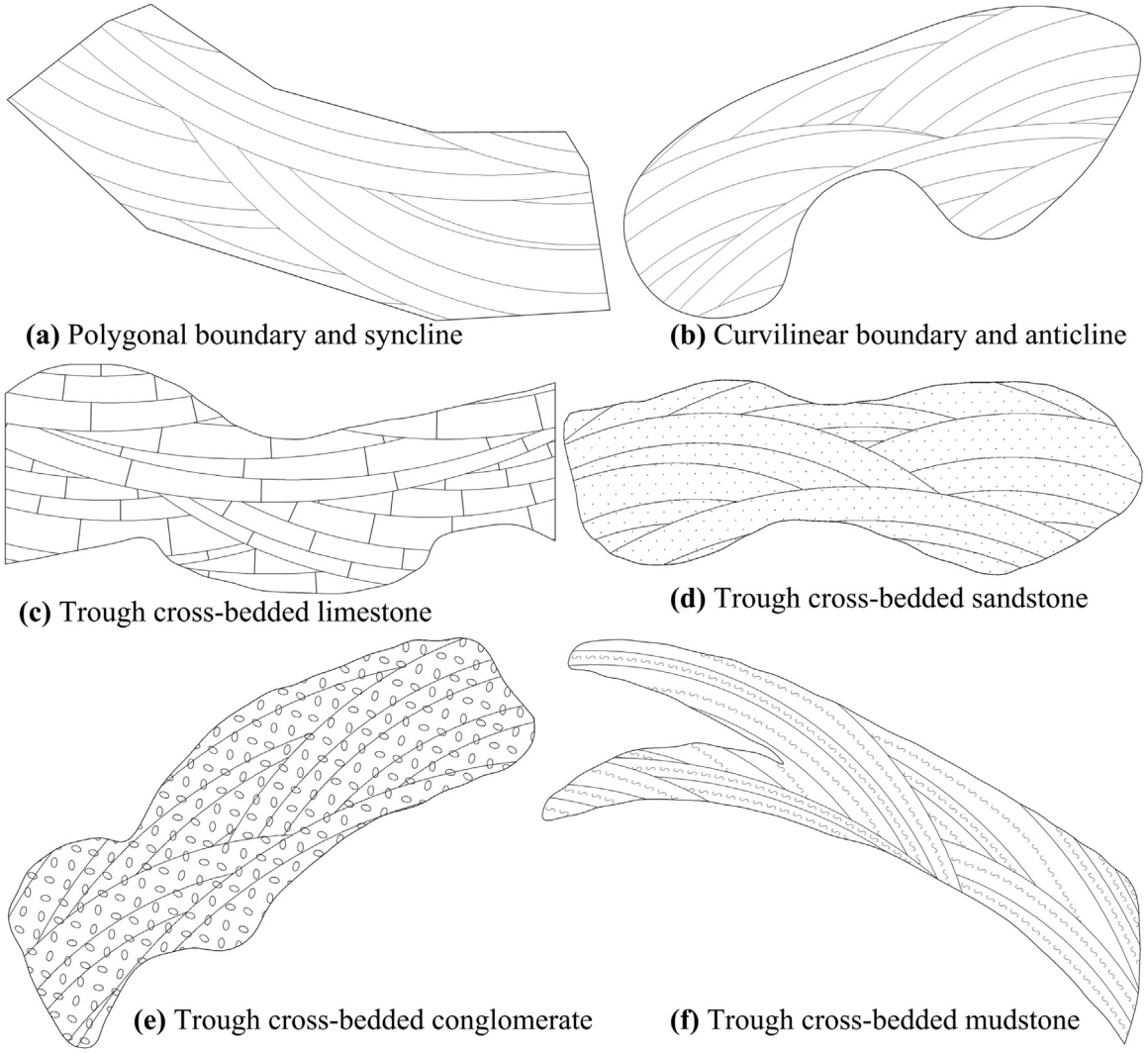


**Fig. 13.** Trough cross-bedding patterns generated by this method and its extended application.

rather than the whole fractal pattern is order to meet the shape features of patterns, cartographic standards and other conditions better. The automatic generation of whole fractal patterns and taking them into the digital mapping based on geospatial data (Jiang, 2015) are our further work in the future.

## 5. Conclusions

The paper presented a method for the automatic drawing of random planar patterns and symbols based on a fractal generator, which aims to address an ongoing challenge in the field of digital mapping. In this method we developed workflows and algorithms to embed the characteristics of random planar patterns into the deployment of the fractal theory and the Iterated Function Systems. The simultaneous representation of the similarity and randomness of random planar patterns was a key issue in this work, and was realized in the fractal generator by adding random disturbance factors in the Iterated Function Systems. We took two typical random planar patterns, the marshland and the trough cross-bedding, as case studies to test the developed method. The implementation of the method in a software program allowed flexible control of parameters, and generated various marshland and trough cross-bedding patterns that conform to national and community cartographic standards. Results of the two case studies prove the effectiveness of the method, and also show that the presented method is ready to be reused or adapted in software programs for digital mapping. Our work is a beneficial supplement to the technologies in digital cartography.

## Acknowledgments

We are grateful to Professor Mariethoz and two anonymous reviewers for their insightful comments and suggestions which led to the improvements in the manuscript. This work was supported by the Natural Science Foundation of China (41172300, 41572314, 41101368), the National High-tech R & D Program of China (863 Program) (2012AA121401) and the Open Fund of Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences (KLIGIP201504).

## References

Barnsley, M., Hutchinson, J., Stenflo, O., 2005. A fractal valued random iteration algorithm and fractal hierarchy. Fractals-Complex Geom. Patterns Scaling Nat. Soc. 13 (2), 111–146.

Batty, M., 1995. Fractals -new ways of looking at cities. Nature 377, (574-574).

Carlson, C.A., 1991. Spatial-distribution of ore-deposits. Geology 19 (2), 111–114.

Chen, G., Cheng, Q., Zhang, H., 2016. Matched filtering method for separating magnetic anomaly using fractal model. Comput. Geosci. 90, 179–188.

Chen, N., Hao, D., Tang, M., 2009. Automatic generation of symmetric IFSs contracted in the hyperbolic plane. Chaos, Solitons Fractals 41 (2), 829–842.

Cheng, Q., Agterberg, F.P., 2009. Singularity analysis of ore-mineral and toxic trace elements in stream sediments. Comput. Geosci. 35 (2), 234–244.

Cheng, Q., Russell, H., Sharpe, D., Kenny, F., Qin, P., 2001. GIS-based statistical and fractal/multifractal analysis of surface stream patterns in the oak ridges moraine. Comput. Geosci. 27 (5), 513–526.

Darmanto, T., Suwardi, I.S., Munir, R., 2013. Animation model of multi-object in fractal form based on partitioned-random iteration algorithm. Procedia Technol. 11, 93–98.

Federal Geographic Data Committee, 2006. [prepared for the Federal Geographic Data Committee by the U.S. Geological Survey]. FGDC Digital Cartographic Standard for Geologic Map Symbolization: Reston, VA, Federal Geographic Data Committee Document Number FGDC-STD-013-2006, 290 pp. A-37-1, 2 plates.

García-Morales, V., 2016. Fractal surfaces from simple arithmetic operations. Phys. A: Stat. Mech. its Appl. 447, 535–544.

GB/T 14538-93, 1993. Legend and color standard for comprehensive hydrogeologic map. The State Standard of the People's Republic of China. State General Administration of the People's Republic of China for Quality Supervision, Beijing,49 pp. 29.

Gentil, C., Neveu, M., 2013. Mixed-aspect fractal surfaces. Comput.-Aided Des. 45 (2), 432–439.

Głażewski, A., Kowalski, P.J., Olszewski, R., Bac-Bronowicz, J., 2010. New approach to multi scale cartographic modelling of reference and thematic databases in Poland. In: Gartner, G., Ortag, F. (Eds.), Cartography in Central and Eastern Europe: CEE 2009. Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 89–106.

Gustavsson, M., Kolstrup, E., Seijmonsbergen, A.C., 2006. A new symbol-and-GIS based detailed geomorphological mapping system: renewal of a scientific discipline for understanding landscape development. Geomorphology 77 (1–2), 90–111.

Hoelzel, M., 2004. StratDraw: automatic generation of stratigraphic sections from tabulated field data. Comput. Geosci. 30 (7), 785–789.

Jiang, B., 2015. Head/tail breaks for visualization of city structure and dynamics. Cities 43, 69–77.

Kappraff, J., 1986. The geometry of coastlines: a study in fractals. Comput. Math. Appl. 12B (3–4), 655–671.

Kraak, M.J., Ormeling, F., 2010. Cartography: Visualization of Geospatial Data 3rd ed.. Pearson Education Ltd, England, 249.

Kya, B., Yang, Y., 2001. Optimization of fractal iterated function system (IFS) with probability and fractal image generation. J. Univ. Sci. Technol. Beijing 8 (2), 152–156.

Li, Q., Su, D., Li, H., Liu, H., Sun, L., 2009. Approach to general data model of GIS symbol library and symbol library data exchange XML schema. Geo-Spat. Inf. Sci. 12 (4), 235–242.

Longley, P.A., Mesev, V., 2002. Measurement of density gradients and space-filling in urban systems. Pap. Reg. Sci. 81 (1), 1–28.

Lovejoy, S., Schertzer, D., 2007. Scaling and multifractal fields in the solid earth and topography. Nonlinear Process. Geophys. 14 (4), 465–502.

Ma, X., Wu, C., Carranza, E.J.M., Schetselaar, E.M., van der Meer, F.D., Liu, G., Wang, X., Zhang, X., 2010. Development of a controlled vocabulary for semantic interoperability of mineral exploration geodata for mining projects. Comput. Geosci. 36 (12), 1512–1522.

Ma, X., Carranza, E.J.M., Wu, C., van der Meer, F.D., 2012. Ontology-aided annotation, visualization and generalization of geological time scale information from online geological map services. Comput. Geosci. 40 (3), 107–119.

Mandelbrot, B.B., 1967. How long is the coast of Britain. Science 156, 636–638.

Mandelbrot, B.B., 2004. Selected topics in mathematics, physics, and finance originating in fractal geometry, In: Novak, M.M. (Ed.), In: Proceedings of the 8th International Conference on Thinking in Patterns -Fractals and Related Phenomena in Nature, Vancouver, Canada, pp. 1–33.

Martyn, T., 2003. Tight bounding ball for affine IFS attractor. Comput. Graph. 27 (4), 535–552.

Mihalynuk, M.G., Mallory, S.M.S., Grant, B., 2006. Geological symbol set for manifold® geographic information system. Comput. Geosci. 32 (8), 1228–1233.

Moriyama, O., Murata, K., Matsushita, M., 1997. Fractal structure and statistics of growing patterns with competition. fractals - an Interdisciplinary. J. Complex Geom. Nat. 5 (2), 247–265.

Nass, A., van Gasselt, S., Jaumann, R., Asche, H., 2011. Implementation of cartographic symbols for planetary mapping in geographic information systems. Planet. Space Sci. 59, 1255–1264.

Nayef, N., Breuel, T.M., 2013. Building a symbol library from technical drawings by identifying repeating patterns. In: Kwon, Y., Ogier, J. (Eds.), Graphics Recognition: New Trends and Challenges In: Proceedings of the 9th International Workshop, Seoul, South Korea, pp. 69–78.

Qiu, J., Song, W., Jiang, C., Wu, H., Dong, R.M., 2013. CGDK: an extensible CorelDRAW VBA program for geological drafting. Comput. Geosci. 51 (2), 34–48.

Siddiqi, S.S., Idrees, U., Rehan, K., 2014. Generation of fractal curves and surfaces using ternary 4-point interpolatory subdivision scheme. Appl. Math. Comput. 246, 210–220.

Subhakar, D., Chandrasekhar, E., 2016. Reservoir characterization using multifractal detrended fluctuation analysis of geophysical well-log data. Phys. A: Stat. Mech. its Appl. 445, 57–65.

Tarboton, D.G., 1996. Fractal river networks, Horton's laws and Tokunaga cyclicity. J. Hydrol. 187 (1–2), 105–117.

Taylor, D.R.F., 2005. The theory and practice of cybercartography: an introduction. In: Taylor, D.R.F. (Ed.), Cybercartography: Theory and Practice. Elsevier, Amsterdam, 1–13.

Turcotte, D.L., 2005. Applications of power-law (fractal) distributions in geology. In: Cheng, Q., BonhamCarter, G. (Eds.), In: Proceedings of the Annual Conference of the International Association for Mathematical Geology, Toronto, Canada, pp. 36–36.

Wang, C., Zhao, Q., Feng, W., 2015. Low-observable target detection in sea clutter based on the adaptive 3D-IFS algorithm. Opt. - Int. J. Light Electron Opt. 126 (20), 2464–2469.

Zhou, J., Li, P., 2008. Development and application of coastline fractal interpolation computing system based on GIS and fractal theory. In: Proceedings of the 6th IEEE International Conference on Industrial Informatics, Daejeon, South Korea, pp. 48–53.

Zhou, Y., Fung, T., Leung, Y., 2016. Improved triangular prism methods for fractal analysis of remotely sensed images. Comput. Geosci. 90 (5), 64–77.