The Visual Computer (2019) 35:223–237
https://doi.org/10.1007/s00371-017-1465-7

ORIGINAL ARTICLE

Procedural modeling of rivers from single image toward natural scene production

Jian Zhang[1] · Chang-bo Wang[1] · Hong Qin[2] · Yi Chen[1] · Yan Gao[1]

Published online: 28 December 2017

Abstract
The rapid and flexible design of natural environments is an important yet challenging task in graphics simulation, virtual reality, and video game productions. This is particularly difficult for natural river modeling due to its complex topology, geometric diversity, and its natural interaction with the complicated terrain. In this paper, we introduce an integrated method for example-based procedural modeling to overcome such difficulties. First, we propose a compact parametric model to represent the certain river, which inherits typical features of natural rivers such as tributary, distributary, tortuosity, possible lakes adjacent to the river. Then, we demonstrate our method for generating 3D river scene solely based on the parametric model. However, choosing appropriate parameters is a tedious undertaking in practice. To further enhance our method's functionality, we rely upon a natural river image to extract meaningful parameters toward the rapid procedural production of the new river scene. Finally, we design a new method to compare two river scenes and iteratively optimize the river network by using the simulated annealing technique. Our method can produce natural river scenes from an example river network and single terrain image with little interaction, and the synthesized scene is visually consistent with the input example in terms of feature similarity. We also demonstrate that our procedural modeling approach is highly automatic toward rapid scene production through various graphics examples.

Keywords Procedural modeling · Natural river generation from images · Natural phenomena

1 Introduction

Conventional 3D modeling still remains a tedious task especially for novice users, in spite of the rapid proliferation of various modeling software in recent years. This is especially the case when we intend to model large-scale outdoor scenes involving natural landscape and/or urban structures with many variations. In reality, we wish to have automatic or semi-automatic techniques toward the rapid creation of 3D digital contents, which promise to reduce a lot of tedious works by users. For decades, procedural modeling has been applied to handle many problems of similar kinds such as terrains, roads, trees, villages, or even cityscape [7,19,23,29,32,34]. In essence, procedural modeling is a generative content creation technique using a program or grammar. A wide variety of plausible 3D digital contents can be created automatically if a grammar is properly defined.

Rivers are very common in natural scenes and play an essential role to enrich users' experiences during game playing or film watching, and the existence of rivers on any type of terrain also offers a beautiful scenery with various settings. Nevertheless, modeling curved river on an existing terrain is quite difficult due to its complex topology, abundant types of shape geometry and its natural adaptation with terrain constraints. A traditional solution is to manually create river scenes by using professional modeling tools, aided by diversified meshes, particles, and animated textures. At the same time, the user should maintain the consistency between rivers and terrains, as well as the self-consistency of the river network, which is a long, laborious, and tedious work even for professional users. In recent years, researchers have also made progress toward production of modeling for

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00371-017-1465-7) contains supplementary material, which is available to authorized users.

✉ Chang-bo Wang
cbwangcg@gmail.com; cbwang@sei.ecnu.edu.cn

[1] School of Computer Science and Software Engineering, East China Normal University, Putuo District, Shanghai, China

[2] Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA

water bodies including rivers. Procedural methods toward river scenes [4,14,42] can generate diverse plausible results, but existing techniques are generally lack of controllability. Interactive methods [8,9] produce controllable river scene, but the user should always deal with fussy work in order to keep the consistency between rivers and terrains. We propose an inverse procedural approach to the synthesis of river scenes, which extracts features from exemplars. The inverse method offers a rapid creation of 3D contents while also considering the user's intent, which could save much labor time.

How to strike a balance between the user control and user interaction is a delicate undertaking for the river scene modeling. This paper aims at solving this problem by addressing two aspects. First, we should find a powerful parametric model to represent a river network, which catches features well such as tributary, distributary, tortuosity, and possible lakes adjacent to the river nearby. The process of expanding a river network should also be designed carefully so that we can generate the river associated with terrain while affording flexible user control. Second, the result of the procedural model should be iteratively optimized due to the randomness that may be caused by procedural processes.

In this paper, we present a novel procedural method to generate the curved river, with features similar to the given example, while still keeping the river network consistency with the terrain and itself. The individual river segment is generated by using a midpoint displacement method subject to terrain consistency. We expand the whole river network on the terrain from the parametric model. The input to our method consists of the river image from a real scene and a terrain over which the modeled river will flow. Our method represents the input river scene image as a directed acyclic graph and extracts a set of parameters (to be detailed in Sect. 4), which is then used to regenerate similar river scene upon new terrain. Our main contributions are as follows:

- We design a novel compact parametric model for representing a river scene and a procedural method to generate various synthetic river scene corresponding to a terrain. Our method for river modeling produces various results and keeps the consistency between river and terrain.
- We introduce a method for extracting shape features of the river network from single image through user interaction. This process is meaningful and can reduce the complexity of appropriate parameter selection for river scene.
- The result generated by our parametric model is still not so controllable since the method is stochastic and the river network is affected by terrain. We then devise a method for similarity evaluation of two river scenes and a method to iteratively optimize a given river network.

The remainder of this paper is organized as follows. Section 2 briefly reviews related works. We provide the overview of our method in Sect. 3. Section 4 describes our parametric model for a river scene, and how we generate a river network on certain terrain. Section 5 details how the result river scene is generated from the input river image. Section 6 demonstrates the implementation details and the results, with a brief discussion on limitations of our method, and Sect. 7 concludes the paper with possible future work.

2 Related works

To the best of our knowledge, the similarity computation of two river scenes and reverse modeling of rivers have not been addressed by the computer graphics community. Our work is related to procedural approaches and interactive methods of rivers. We also briefly review example-based methods because of their great relevance to this paper.

Procedural modeling Procedural techniques have been used in computer graphics for a long time, and several authors have proposed methods for river modeling on terrain. Despite that the results are always hard to control, we benefit from classical procedural methods which combine fast production and plausible results. Kelley et al. [22] first presented a procedural method that generates river network with terrain scene. They used a recursive algorithm to branch and subdivide a single river path, in order to create a river network. Prusinkiewicz et al. [30] introduced a method which combines context-sensitive geometric rewriting and midpoint displacement to generate rivers in mountains. Derzapf et al. [6] produced river network at a planetary scale. They generated adaptively refined scene geometry during flythrough by exploiting current graphics hardware, starting with a coarse geometry of a planet. Procedural methods based on hydrology [13,25,36,42] offer rapid production of river scene, but they are lack of user control. Génevaux et al. [14] combined feature-based primitives and hierarchical procedural modeling to generate terrain scene including rivers. Recently, Cordonnier et al. [4] introduced a method to produce landscapes with high-level control from uplift map, which combines uplift and hydraulic erosion.

Interactive methods Interactive methods with sketching [20,35] have also been used in river scene modeling, but these methods do not always guarantee physical plausibility. Yu et al. [41] presented a method for interactive simulation of running fluids, which provides a plausible rendering result. Using 3D curves to control the shape of terrain and water body, Hnaidi et al. [17] proposed an interactive method based on diffusion equation. Emilien et al. [8] described a framework of generating waterfall scene, combining procedural generation and user control. Samavati et al. [33] proposed their method for adding terrain features as well as water bodies to Digital Earth, which combines sketching and 3D reconstruction. Compared with these approaches, our method needs less user interaction.

Example-based modeling Our method is closer to example-based modeling, which generates similar objects from examples [26]. They are widely used in texture synthesis [16,21,24]. Example-based methods also have been applied to generate 3D geometric objects [11], trees [37,38], indoor scenes [10,40], terrain scenes [2,43], and buildings [1,15]. Emilien et al. [9] introduced a method for interactive synthesis of virtual landscapes based on examples, but their method relies on plenty of user operations. Recently, more and more authors combined example-based procedural modeling with machine learning [28,39]. Nishida et al. [27] presented a method that allows users to design road network by specifying road examples, which inspires this research to explore along a similar direction toward river scene production. They extracted patches from examples and grew new road network using these patches while combining procedural approaches. But for a river scene we need to consider more about the coupling between river and terrain.

3 Method overview

To find the procedural representation for various kinds of rivers, we need a powerful procedural model to create river scenes. In Sect. 4.1 we depict our compact parametric model for rivers, divided into two categories: top-down parameters and cross-sectional parameters. Given the starting and ending positions, our method expands the river network by using the rules described in Sect. 4.2, while maintains the consistency between river and terrain. Our procedural rules allow the generation of distributaries, tributaries, and lakes in the river scene. Nevertheless, due to the complexity of choosing appropriate parameters, we then seek a new solution of extracting parameters from a single image.

An overview of the pipeline of our method is shown in Fig. 1. First, the user selects a river image which is to be analyzed and provides terrain height map which is used to generate terrain in the final scene. Our input of image is the aerial view of a river scene such as an aerial photograph or picture extracted from available geographic information systems. To extract the top-down features of a given river, the user should specify the start position and the end position of the river network and we then represent the input river as a directed acyclic graph with some parameters (detailed in Sect. 5.1).

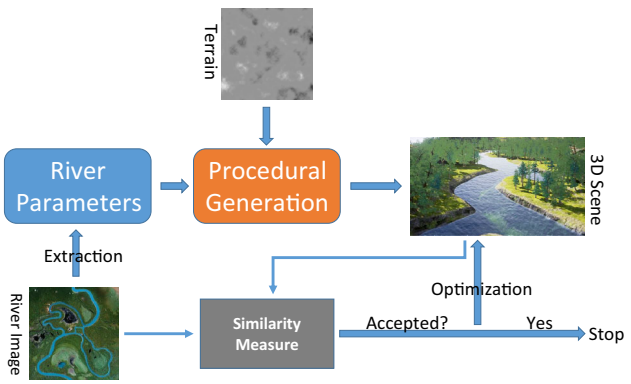


Fig. 1 Framework overview: the input river image is used to extract river parameters and helps the generation of river network upon new terrain. Final synthesized scene is generated via iterative optimization

Once the top-down parameters of the river image are extracted, the initial river network can be created by our procedural model, with the user-provided terrain and cross-sectional parameters. Then we start the optimization process (Sect. 5.3), which is necessary because of the top-down difference between the input image of the river and the result river. We use a simulated annealing framework to iteratively modify the river network and finally get a convergence result. Our method for similarity computation between two river networks intends to compare the top-down features of two river networks (described in Sect. 5.2). At the end of each iteration, the optimization process could be terminated if the similarity between two river scenes reaches the threshold value.

4 Procedural model for river

Rivers on different terrain have many special features that can be represented by a powerful procedural model. Inspired by [13], we introduce a procedural method to expand river network. Unlike the method depicted in [13], where they created the river tributary by iteratively dividing one river segment to two from downstream to upstream, our model is capable of generating several typical characteristics such as distributary, tributary, and connected lake in river scene. In this paper we call that, a distributary is a river that flows from mainstream, and a tributary is a river that flows into mainstream (see Fig. 2 for a graphical illustration).

4.1 Parameters of river network

To generate various river scenes, we need to find out the common characteristics and categorize them into a set of parameters that can be used in programming. By studying the existing river procedural model and looking at many real scenes, one compact parametric model is proposed. Our parametric model is designed to be used in the river scene generation step. The parameters comprise two parts: top-down parameters and cross-sectional parameters. The top-down parameters control the characteristics of river network such as tortuosity of rivers and distributary number, which contribute to the diversity of river network. On the other hand, the cross-sectional parameters define the features that are responsible for the integration of mesh generation in

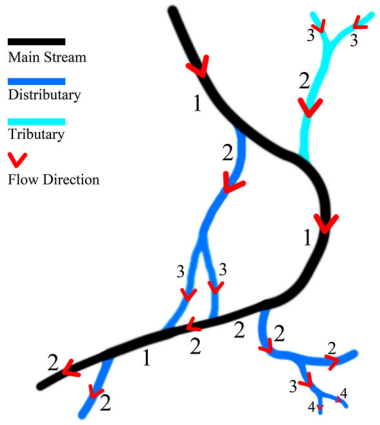


Fig. 2 The level of river segment used in our method. At the source of a river, the segment is marked as level one. The level of distributaries and tributaries should be greater than or equal to the level of their parent river. If the level marking is conflicted, we select the larger one. In this figure, there is two isolated regions because of the distributaries flowing into the river network

Table 1 Our parameters for river generation

Type	Parameters	Description
Top-down	n_r	The number of single river segment
	d_r	The straight line distance from river start to river end
	k_r	The ratio of the actual length of river to the straight distance of river
	k_{sm}, k_{sv}	The mean and variance of the bending factor for single river segment
	d_{sm}, d_{sv}	The mean and variance of the straight line distance for river segment
	n_d	The number of river distributary
	n_t	The number of river tributary
	n_c	The number of isolated regions
	θ_{bm}, θ_{bv}	The mean and variance of the angle between the direction of one river segment and its distributary
	s_{lm}, s_{lv}	The mean and variance of the area of lake
	n_l	The number of lake in river scene
Cross section	w_r, χ_{rw}	The river width for mainstream and decrease factor for smaller level river
	h_r, χ_{rh}	The river depth for mainstream and decrease factor for smaller level river
	φ	Controls whether a river increases level when it bifurcates
	h_{lm}, h_{lv}	The mean and variance of the river lake

river scene and also influence terrain, which are not extracted from input river image. The parameters are shown in Table 1.

Top-down parameters represent attributes of diverse river networks. n_r defines the number of river segment, which is represented as an edge in our directed acyclic graph. n_d and n_t define the number of river distributaries and tributaries, respectively. A distributary contributes to an island surrounded by rivers when it flows into a river segment (Fig. 2) and we use n_c to count the number of the isolated regions. The initial flow direction of a river segment is determined by the angle θ; $\{\theta \sim N(\theta_{bm}, \theta_{bv})\}$, where N is the normal distribution, with θ_{bm} the expectation and θ_{bv} the variance. Tortuosity is a vital property for river segment. For a single river segment, we define the tortuosity as the ratio of the actual river length to the straight line distance of the river. We bring in parameter k_r to represent the tortuosity of the whole river network. For every river segment, the straight line distance of river is sampled from the normal distribution with expectation d_{sm} and variance d_{sv}, and the tortuosity is also determined by the parameter k_{sm} and k_{sv}.

Cross-sectional parameters are used in the generation of integrated meshes of the river and the adapted terrain. In our framework, the river segment is labeled by a number of level (Fig. 2), the river with a bigger number often has a smaller water body. χ_{rw} and χ_{rh} affect the decrease speed of river width and depth, respectively. When we expand the river network, the newly added distributary or tributary has an incremental level with probability φ.

4.2 River network generation

In our framework, the river network growth is restricted because users specify the start and end position of the river, while the altitude of river source should be greater than the river mouth. A river may not arrive at the specified position finally if it flows freely. To simplify the generation, we solve the mainstream first and create distributaries and lakes then. Every watercourse of the river segment is represented as a B-spline curve.

4.2.1 Mainstream

The river is generally curved and smooth, which can be fitted by a B-spline curve. By using the midpoint displacement, we alter the data points of B-spline and recalculate the watercourse of a river, while also maintaining the consistency between rivers and terrain. Our algorithm starts with a straight line segment from river's start position to its end position. Then the line segment is iteratively divided to $n_r - n_d - n_t$ parts (Fig. 3). At last the algorithm recursively modifies the segment of the mainstream to make the river fitting the terrain. Figure 4 shows how we modify a river segment at first iteration, and we also iteratively reduce the value of d in the same way as that in the traditional approach, which makes the river more smoothly. Examine Fig. 4, the curved river should flow with the terrain and the constraint problem is formulated as:

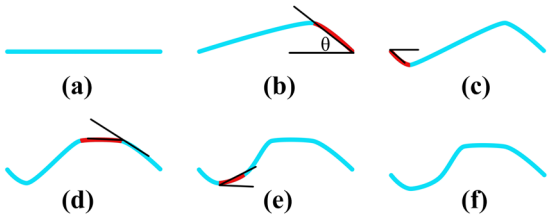


Fig. 3 The mainstream is iteratively divided into five segments. At every iteration, the length of the displaced segment (colored in red) is sampled from normal distribution $N(d_{\text{sm}}, d_{\text{sv}})$ and the branch angle θ is sampled from normal distribution $N(\theta_{\text{bm}}, \theta_{\text{bv}})$

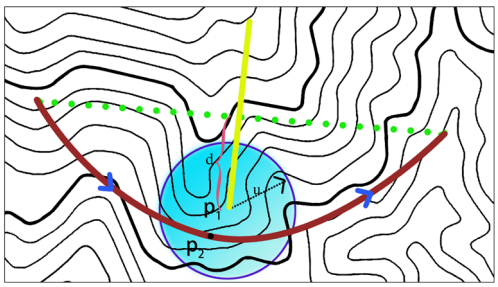


Fig. 4 To apply midpoint displacement, we first select a point $\mathbf{p_1}$ at the perpendicular bisector of the river segment. We select $\mathbf{p_2}$ by calculating Formula 1 at the range $\mathbf{p} \in \text{circle}_{\mathbf{p_1}}$. The distance d between $\mathbf{p_1}$ to the original river is sampled according to k_{sm} and k_{sv} (Table 1), where u is the radius of $\text{circle}_{\mathbf{p_1}}$ that we set it to be half of d. That is to say, we select $\mathbf{p_1}$ which makes the length of this curve in agreement with the sampled tortuosity

$$\underset{\mathbf{p}}{\text{argmin}} \left(\int_e \text{clamp}\left(\frac{\partial h}{\partial r}, 0, \infty \right) \right), \tag{1}$$

$$\text{clamp}(x, a, b) = \begin{cases} a, & x < a \\ x, & a \le x \le b, \\ b, & x > b \end{cases} \tag{2}$$

where $\frac{\partial h}{\partial r}$ indicates the directional derivative of the terrain height along the direction of the river e. We solve Formula 1 by using a stochastic sampling method and $\mathbf{p_2}$ is selected in the range $\mathbf{p} \in circle_{\mathbf{p_1}}$. Note that, we clamp the gradient of terrain height in the range $(0, \infty)$, because we only punish that rivers flow from low altitude to high altitude.

4.2.2 River network

To expand the whole river network $\mathcal{R}$, we need a marking rule for the river segment. Inspired from Horton–Strahler number [18], we design our marking rule shown in Fig. 2. As showcased in Fig. 2, the river segment from the river source is marked as level one, while it increases with a probability when it bifurcates to a distributary and decreases when meets a tributary. A larger level number indicates the river is narrower and shallower. We put forward that a river network $\mathcal{R}$ is defined by a set of nodes $\{\mathcal{N}\}$ and a set of river segments $\{E\}$. Every river segment e_i has the branch angle θ_i (if it has a parent river segment), bending factor k_i, straight line distance d_i, the start node $n_{i,1}$, and the end node $n_{i,2}$.

Let $\Upsilon^* = \{n_{\text{r}}^*, d_{\text{r}}^*, k_{\text{r}}^*, ...\}$ denote the desired values of the river parameters. Algorithm 1 describes how we expand a river network with the mainstream and the desired parameters Υ^*. As showcased in Algorithm 1, a river segment is selected and could be expanded until there are enough distributaries, tributaries, isolated regions, and lakes.

Algorithm 1: River Network Expansion

Input: a input mainstream $\mathcal{R}$, desired parameters Υ^*
Output: whole river network $\mathcal{R}$

$n_{\text{d}}^{'} \leftarrow 0, n_{\text{t}}^{'} \leftarrow 0, n_{\text{l}}^{'} \leftarrow 0, n_{\text{c}}^{'} \leftarrow 0$
while $n_{\text{d}}^{'} < (n_{\text{d}}^* - n_{\text{c}}^*)$ **do**
  Select the segment e to expand.
  Compute branch angle θ, bending factor k and straight line distance d from Υ^*.
  if *Execute Rule 1 with* θ, k, d *for segment* e **then**
    $n_{\text{d}}^{'}++$
  end
end
while $n_{\text{t}}^{'} < n_{\text{t}}^*$ **do**
  Select the segment e to expand.
  Compute branch angle θ, bending factor k and straight line distance d from Υ^*.
  if *Execute Rule 2 with* θ, k, d *for segment* e **then**
    $n_{\text{t}}^{'}++$
  end
end
while $n_{\text{c}}^{'} < n_{\text{c}}^*$ **do**
  Select the segment e to expand.
  Compute branch angle θ, bending factor k and straight line distance d from Υ^*.
  if *Execute Rule 1 with* θ, k, d *for segment* e *and create an isolated region* **then**
    $n_{\text{d}}^{'}++, n_{\text{c}}^{'}++$
  end
end
while $n_{\text{l}}^{'} < n_{\text{l}}^*$ **do**
  Select the segment e to place a lake.
  Compute size s from Υ^*.
  if *Execute Rule 3 with* s *for segment* e **then**
    $n_{\text{l}}^{'}++$
  end
end
return $\mathcal{R}$

River segment selection We use a probabilistic method to determine which river segment should be expanded. A river segment e_i cannot be expanded if it already has two extended branches; otherwise, it will be expanded at a probability ζ_i:

$$\zeta_i = \frac{\rho_i}{\sum_{n=1}^{N} \rho_n}, \tag{3}$$

where $\rho_i = g_i + f_i$ is the priority value that takes into account the priority of terrain height g_i and the priority of river level f_i, and g_i is computed as follows:

Table 2 Expansion rules for the river network

1.1	$s(n) \rightarrow s(n)d(n+1) : (\varphi)$
1.2	$s(n) \rightarrow s(n)d(n) : (1-\varphi)$
2.1	$s(n) \rightarrow s(n)t(n+1) : (\varphi)$
2.2	$s(n) \rightarrow s(n)t(n) : (1-\varphi)$
3	$s(n) \rightarrow s(n)l$
4	$s(n) : Distance(s(n), \mathcal{R}) < \delta \rightarrow s^{'}(n)$
5	*Midpoint Displacement*$(s(n))$

$$g_i = \frac{\tau_c - \tau_{\min}}{\tau_{\max} - \tau_{\min}}, \tag{4}$$

where τ_c is the height of the river segment's end node $\mathcal{N}_{i,2}$ and $\tau_{\max}$ the maximum height in $\mathcal{R}$ and $\tau_{\min}$ the minimum height. f_i is computed as follows:

$$f_i = 1 - \frac{l_i - l_{\min}}{l_{\max} - l_{\min}}, \tag{5}$$

where $l_{\max}$ is the maximum level in $\mathcal{R}$ and $l_{\min}$ the minimum level.

Lakes are randomly placed on the nodes $\{\mathcal{N}\}$ in our framework. Besides, lakes are preferred to be placed on the junctions of river segments because in the real world lakes are usually not at the endpoints of a river.

River segment expansion Each river segment is expanded according to the pre-defined parameters. Our rules of river expansion are defined in the following form:

predecessor : *condition* $\rightarrow$ *successor* : *probability*, where *predecessor* is a river segment that is to be expanded to *successor* (*predecessor* $\in \mathcal{R}$). The rule is selected with *probability* when *condition* is met.

The river network is expanded using the rules described in Table 2, where $s(n)$ defines one river segment with level n, and $d(n)$ and $t(n)$ define distributary and tributary, respectively. Rule 1–3 expand a distributary, a tributary, and a lake, respectively. When expanding distributary, Rule 1.1 is selected with probability φ (Table 1), which is the same as Rule 2.1 when expanding tributary. In Algorithm 1, Rule 4 and Rule 5 are executed when any distributary or tributary is successfully added to $\mathcal{R}$. We extend a river segment s to the river network $\mathcal{R}$ when its tail is closed to $\mathcal{R}$ (Rule 4). Lakes are added to river network by using Rule 3. When expanding rules are executed, the branch angle, the straight line distance, etc., are sampled from normal distribution depending on our parametric model described in Sect. 4.1. Several types of distributaries are demonstrated in Fig. 5.

While expanding a new distributary e^* to a river segment e, the segment e^* flows from e to a low elevation area. In contrast, e^* flows from higher place to e when expanding a new tributary.

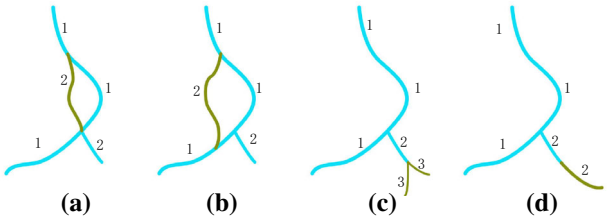


Fig. 5 Different types of distributaries, the number representing the river level: **a** a distributary meeting existing river node and generating a isolated region; **b** a distributary meeting river segment and creating new river node; **c** two distributaries creating from one river segment; **d** a continuation of a river segment

While expanding a new distributary e^* to a river segment e and creating an isolated region, the start node n_1 of e^* is set to be the end node of e. Recall that in our framework the structure of a river is a directed acyclic figure that can be sorted by topology. Let $\{\mathcal{N}^{'}\}$ denote a subset of $\{\mathcal{N}\}$ in which the topology order of any node is no greater than n_1. In $\{\mathcal{N}^{'}\}$ we select a node n^* that should meet the formula as follows:

$$\min\left(|Distance\left(n_1, n^*\right) - d|\right) < \phi, \tag{6}$$

where d defines the desired value of straight line distance sampled from Υ^* and ϕ is the threshold value defined by users. Then we set n^* as the end node of e^* based on Eq. 6.

Collisions avoidance Collisions could occur when we add a distributary or tributary to a river network, or when we bend a river segment to make it fit a terrain. At the expansion step we use a two-dimensional mask to store the location of the river network on terrain. A new river segment is added to the river network only if there is no collision.

4.3 Example of river scenes

Figure 6 shows three river scene generated using our framework, demonstrating the diversity of our parametric model. The parameter values in Table 3 affect the result a lot, while another impact is the terrain, because our midpoint displacement method is executed along with the terrain. Handling these 22 parameters is somehow tedious for the river scene generation, so we seek a method that could extract parameters directly from real river image.

5 River generation from image

In this section, we discuss the generation of river scene from river image and existing terrain. In our framework, the user provides a river scene image, and we extract top-down parameters (Table 1) from the image after a simple interaction (Fig. 7). Then by using our procedural method, we gener-

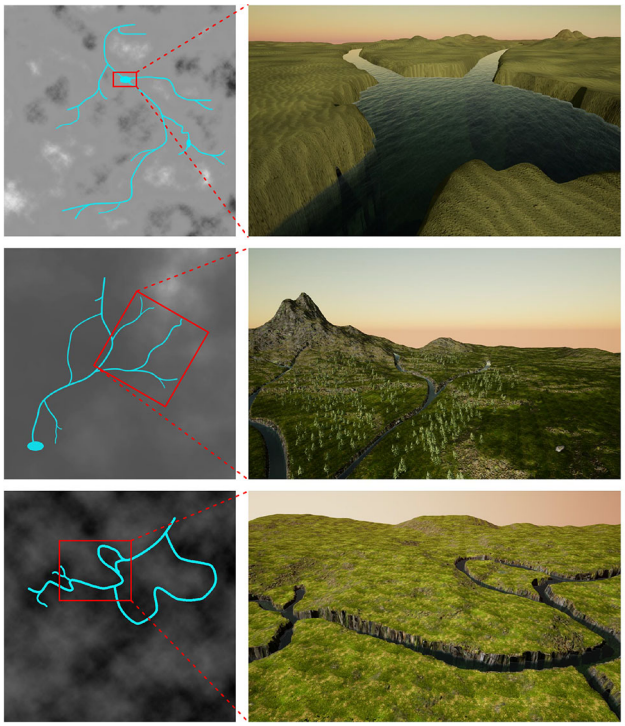


Fig. 6 Different river scenes generated by our system. On the left are the 2D views of the scenes, and the right are 3D scenes. The corresponding values of parameters are in Table 3

Table 3 Our parameters for river generation

Parameters	Figure 6a	Figure 6b	Figure 6c
n_r	25	20	13
d_r	3120	2177	2103
k_r	2.76	2.93	2.61
k_{sm}	1.28	1.31	1.76
k_{sv}	0.22^2	0.21^2	0.78^2
d_{sm}	643	667	562
d_{sv}	483^2	511^2	415^2
n_d	17	5	6
n_t	0	8	0
n_c	0	1	2
θ_{bm}	0.78	0.77	1.12
θ_{bv}	0.18^2	0.44^2	0.82^2
s_{lm}	8246	9923	0
s_{lv}	3714^2	0	0
n_l	2	1	0
w_r	28	32	43
χ_{rw}	0.91	0.85	0.95
h_r	15.94	22.72	28.62
χ_{rh}	0.88	0.71	0.93
φ	0.88	0.82	0.91
h_{lm}	48.46	50	0
h_{lv}	20.37^2	0	0

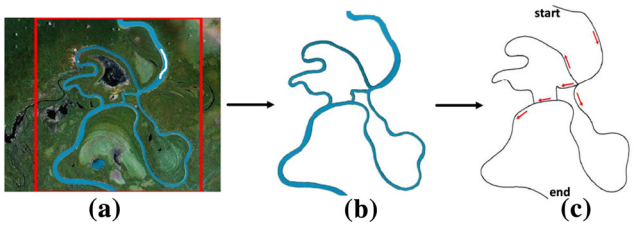


Fig. 7 River network extracted from image: **a** the user drags a rectangle around the river and selects a region of the river with a stroke; **b** the image after segmentation; **c** the river network extracted from the segmentation image with the user specifying the river source and the river mouth

ate an initial river network, with user-defined cross-sectional parameters. The final river network is generated from iterative optimization.

5.1 Parameters learning from image

We have introduced our procedural model for rivers in Sect. 4, yet the result is not easy to control because the number of parameters is large and the method is stochastic. Then we try to extract river features from an image, which is the top view of the river scene. We use Grabcut described in [31] to generate segmentation image for river image. Grabcut affords a good result that extracts the correct contour with boundaries of a river from an image (Fig. 7). The region of a river is segmented with the help of user interaction. Then we extract the skeleton of river network by using the method introduced in [3], afterward each river segment is retrieved by traversing the skeleton of the image. The flow direction of each river segment e is automatically determined based on the assumption that the branch angle $\theta < \pi/2$ (Fig. 8). We re-group

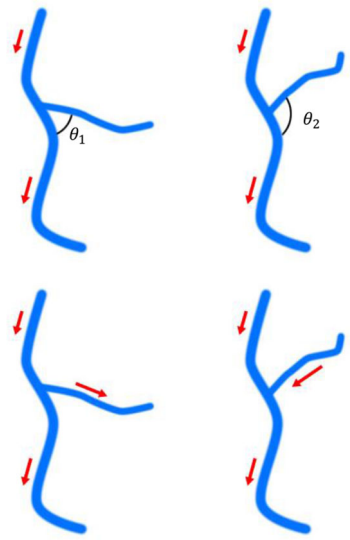


Fig. 8 The flow of river segment is determined with the branch angle ($\theta_1 < \pi/2, \theta_2 > \pi/2$)

these river segments as a directed acyclic graph $\mathcal{R}$, which represents the river network of the input image. After a preprocessing (Algorithm 2) for example river, we can calculate the top-down parameters Υ^* from $\mathcal{R}$. Given a new terrain, a start position p_s and an end position p_e, we could generate a new river network with Υ^*. The river scale parameter μ is introduced to control the size of the generated river:

$$\mu = \frac{Distance(p_s, p_e)}{d_r^*}. \tag{7}$$

Then we multiply $n_r^*, n_d^*, n_t^*, n_l^*$ by μ.

Algorithm 2: Preprocessing for Example River $\mathcal{R}$.

```
compute the shortest path from river source to river month and
  mark them as mainstream {E_m}.
foreach river segment e in R and e ∉ {E_m} do
    if e flows into {E_m} then
        mark e as a tributary
    end
    if e flows from {E_m} then
        mark e as a distributary
    end
end
```

However, as mentioned before, the river scene is also controlled by the terrain, so that this river network may not be our expected result. On the other hand, many operations in our procedural model use sampling, and this may generate ambiguity result when a river is not large. In order to find the optimal river scene fitting the terrain and being similar to the example scene, we design the distance evaluation method and iteratively modify our result.

5.2 Evaluation of river scene production

As the initial result river network has been generated, we need a distance function to evaluate the difference between our result with the input image. Some parameters can be controlled accurately, such as the number of tributaries. We consider the tortuosity distance ξ_c and the branch angle difference ξ_a as follows:

$$\xi_c = 1 - \exp\left(-\left(\left|k'_{sm} - k_{sm}\right| + \left|\sqrt{k'_{sv}} - \sqrt{k_{sv}}\right|\right)\right), \tag{8}$$

$$\xi_a = 1 - \exp\left(-\frac{\left|\theta'_{sm} - \theta_{sm}\right| + \left|\sqrt{\theta'_{sv}} - \sqrt{\theta_{sv}}\right|}{\pi/2}\right), \tag{9}$$

where k'_{sm}, k'_{sv}, θ'_{sm}, and θ'_{sv} are the corresponding parameter values of the generated river network. We also consider another distance evaluation, which is not evident in our parametric model. When a distributary flows into the river network, one piece of land surrounded by rivers appears (Fig. 2). We calculate this feature ξ_l especially:

$$\xi_l = 1 - \exp\left(-\frac{(S(\mathcal{R}_1) - S(\mathcal{R}_2))^2}{S^2(\mathcal{R}_1)}\right), \tag{10}$$

where $S(\mathcal{R}_1)$ and $S(\mathcal{R}_2)$ define the area of isolated region in two river scenes. Then we give the top-down distance function:

$$D(\mathcal{R}_1, \mathcal{R}_2) = \omega_c\xi_c + \omega_a\xi_a + \omega_l\xi_l. \tag{11}$$

The values ω_c, ω_a, and ω_l, satisfying $\omega_c + \omega_a + \omega_l = 1$ are weight coefficients associated with the distance function. The user controls the final production by defining these parameters. Figure 9 and Table 4 document the effect of different parameters on the optimized river network, and demonstrate that the results after optimization are visibly better than that before optimization. The river scene presented in Fig. 9b ($\omega_c = 0.6$, $\omega_a = 0.2$, $\omega_l = 0.2$) shows the approximate tortuosity but more difference in isolated region. In contrast, Fig. 9c ($\omega_c = 0.2$, $\omega_a = 0.2$, $\omega_l = 0.6$) has a nearly equal area of isolated region related to Fig. 7 because the parameter ω_l gives a high importance to isolated region difference, while the tortuosity is not so ideal. In Sect. 5.3 we will discuss the usage of distance function in details.

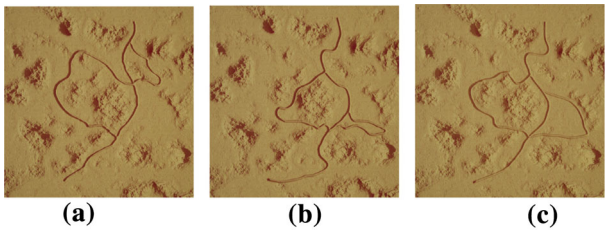


Fig. 9 The generated result without optimization (**a**). Two sets of parameters for top-down distance function can result in diverse output scenes (**b**, **c**). The input river network is shown in Fig. 7

Table 4 The values of top-down distance at the state of convergence (corresponding to Fig. 9)

	ξ_c	ξ_a	ξ_l	Top-down distance
Figure 9a	0.35	0.18	0.23	–
Figure 9b	0.18	0.12	0.29	0.19
Figure 9c	0.31	0.16	0.16	0.19

5.3 River network optimization

We use a simulated annealing framework to optimize the result scene (Algorithm 3). Our algorithm starts with the river

Algorithm 3: River Network Optimization.

Input: Example river network $\mathcal{R}$, Result river network $\mathcal{R}_1$,

1 $t \leftarrow t_0$
/* n: total iteration times */
2 **for** i *from* 1 *to* n **do**
/* m: trial times in one iteration */
3 **for** j *from* 1 *to* m **do**
4 $\mathcal{R}_1^{'} \leftarrow$ perturb $\mathcal{R}_1$
5 **if** $D(\mathcal{R}, \mathcal{R}_1^{'}) < \sigma$ **then**
6 $\mathcal{R}_1 \leftarrow \mathcal{R}_1^{'}$
7 stop
8 **end**
9 $\triangle E \leftarrow D(\mathcal{R}, \mathcal{R}_1^{'}) - D(\mathcal{R}, \mathcal{R}_1)$
10 **if** $\triangle E < 0$ **then**
11 $\mathcal{R}_1 \leftarrow \mathcal{R}_1^{'}$
12 **else if** $rand() < e^{\frac{-\triangle E}{t}}$ **then**
13 $\mathcal{R}_1 \leftarrow \mathcal{R}_1^{'}$
14 **else**
15 continue to next iteration
16 **end**
17 **end**
18 $t \leftarrow t \times \alpha$
19 **end**

network generated from the procedural model, where the top-down parameters are extracted from example river image. The energy function of our annealing framework is based on the distance evaluation described in Sect. 5.2:

$$E = D(\mathcal{R}_1, \mathcal{R}_2), \tag{12}$$

where $\mathcal{R}_1$ defines the example river network and $\mathcal{R}_2$ the current generated river network, respectively. To perform the annealing framework algorithm, we need a perturbation operation (line 4 in Algorithm 3) for the river network. Our perturbation consists of three aspects: river segment, river intersection, and lake. The following discussions describe the perturbation in details:

- $Flex(p_f) : \eta_f(e_1, \mathbf{p_i})$ perturbs the position of river e_1's data point $\mathbf{p_i}$ (not the head or tail), then regenerate the B-spline curve of e_1.
- $Translate(p_t) : \eta_t(e_1, e_{1,i}, e_{1,i,j})$ sets the parent river of e_1 to the adjacent river $e_{1,i,j}$ of e_1's original parent river $e_{1,i}$, and resets the child river (if has) of e_1 as the same way. Regenerate the B-spline curve of e_1 with the same expected tortuosity.
- $Drag(p_d) : \eta_d(n_1, e_{1,i})$ perturbs the position of node of river intersection n_1, and recalculate all the adjacent river $e_{1,i}$ of n_1.
- $Move(p_m) : \eta_m(n_1, n_{1,i})$ moves the position of one lake from node n_1 to the adjacent node $n_{1,i}$ of n_1.
- $Scale(p_s) : \eta_s(n_1)$ changes the size of lake at node n_1.

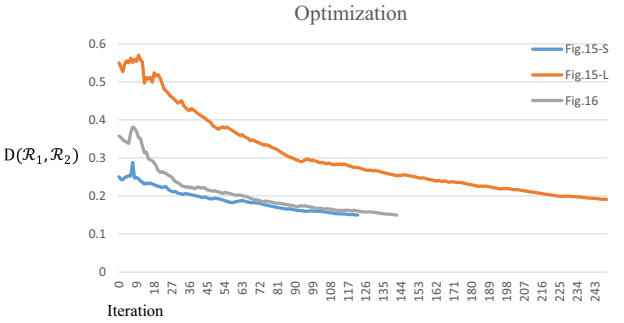


Fig. 10 Optimization of the river scenes. The abscissa and the ordinate represent the iteration times and the value of distance function for two river scenes, respectively. Figure 14-S, -L refers to the smaller and the larger scenes in Fig. 15, respectively

In Algorithm 3, these operations are randomly selected and executed while retaining consistency with terrain and avoiding collisions at the same time. The simulated annealing method operates by iteratively perturbing the current river network, and evaluating where the iteration could be accepted by the current energy. Unlike the classic implementation of the simulated annealing approach, we try m times at every iteration for cooling down temperature so that we get more trials and could find the optimized result. In Algorithm 3, we set $n = 400$, $m = 20$, $t_0 = 0.8$ and the final temperature 0.3. We also set a threshold to the distance function, and the algorithm could stop anytime. The convergence of our algorithm is shown in Fig. 10.

5.4 Guided optimization from sketch

Our optimization algorithm is easy to extend given a second sketch for the river layout. As showcased in Fig. 11, The final river is optimized with the input image and passes through the sketch area. As for sketch guiding, the integrated method is mainly changed in the following three parts:

- Unlike in Eq. 7, we calculate the river length of the sketch as the numerator. The recalculated μ is used to amend parameters Υ^* in the same way as before.
- We analyze the topology of the sketch area and treat it as the main stream. We also bend it to fit the terrain.
- The topology of the sketch area will not be changed at every iteration in Algorithm 3. Besides, the positions of the nodes extracted in the sketch are not changed in the optimization process.

6 Implementation details and experimental results

In this section, we document the implementation details and experimental results generated with our method. First, we

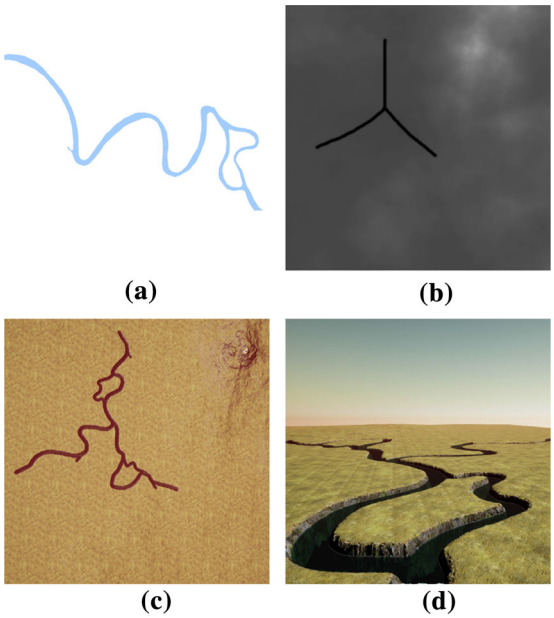


Fig. 11 Guided optimization from sketch: **a** The exemplar river image; **b** A sketch at target terrain; **c** and **d** The generated river network

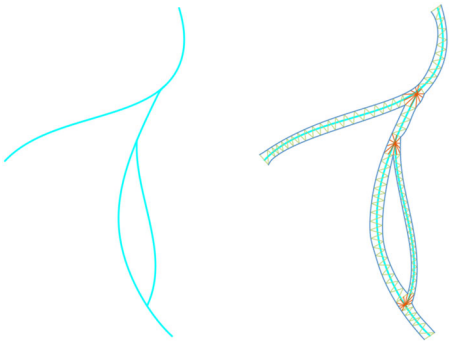


Fig. 12 The schematic of the generated river mesh. On the left is the river network, and on the right is the corresponding mesh. On the left there is a thin tributary because of its greater river level

present the key steps for generating 3D scene from the river network (Sect. 6.1). Then in Sect. 6.2, we present some experiments with evaluation.

6.1 Procedural generation of scenes

3D scene generation mainly consists of two parts: river mesh generation and terrain modification. To ensure the realism of the final 3D results, with impalpable artifacts, we carefully process the river mesh and the terrain.

River mesh generation It may be noted that we use a spline function to smooth the trajectories of the river network. We define the river surface of the whole river network by united 3D meshes (see Fig. 12). Using the trajectories of the river

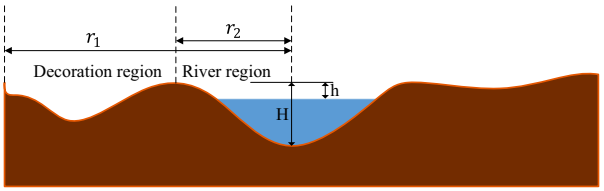


Fig. 13 The region of terrain modification

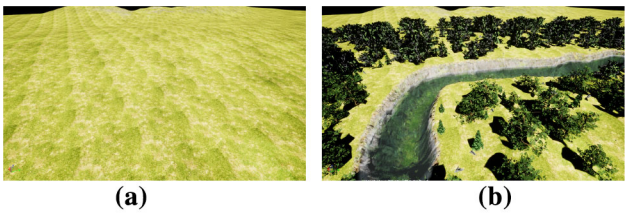


Fig. 14 Terrain before and after deformation and decoration

network, these meshes are generated with a certain width. We calculate the width of a river with level λ by $w = w_r \times (\chi_{rw})^{\lambda-1}$, where w_r defines the width of a river whose level is 1 and χ_{rw} is the decreasing factor of the river width. The intersections of rivers are carefully handled to avoid overlap.

Terrain modification We introduce deformation to the terrain according to the river region (see Fig. 13). The affected terrain region is divided into two parts as follows:

$$\Omega_1 = \{\mathbf{p} \in \Omega \mid Distance(\mathbf{p}, e) < r_2\}, \tag{13}$$

$$\Omega_2 = \{\mathbf{p} \in \Omega \mid r_2 < Distance(\mathbf{p}, e) < r_1\}, \tag{14}$$

where e defines the river trajectory. The corresponding deformation and decoration are performed as follows:

- In Ω_1, excavation is performed precisely to generate the riverbed. Génevaux et al. [13] discussed different types of riverbed depending on water flow, elevation, etc. Our procedural process uses the simple excavation showcased in Hnaidi et al. [17]. The maximum depth H depends on the river level as $H = h_r \times (\chi_{rh})^{\lambda-1}$, where h_r defines the depth of river whose level is 1 and χ_{rh} is the decreasing factor of the river depth, and the distance from final river meshes to the maximum position of the riverbed is set to be h. We also remove the vegetation in the region Ω_1.
- In Ω_2, we add vegetation to decorate the terrain by stochastically planting trees. To smooth the terrain after generating riverbed, we set height h' of position $\mathbf{p}$ as $h' = h_m + (\frac{Distance(\mathbf{p},e)-r_2}{r_1-r_2})^2 \times (h_0 - h_m)$, where h_m and h_0 define the maximum height of the riverbed and the original height at position $\mathbf{p}$, respectively.

The generated river meshes and the modified terrain form the final scene. Figure 14 highlights the comparison between the original terrain and the final river scene.

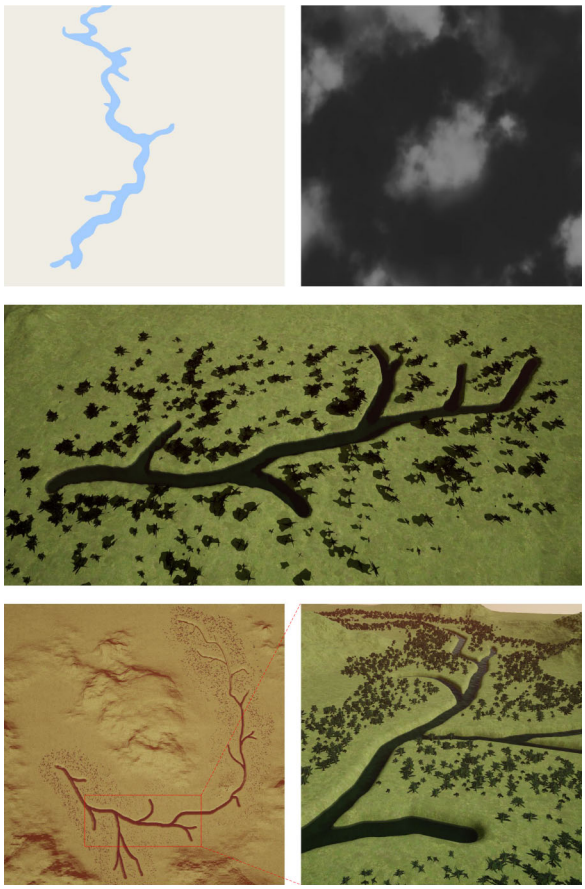


Fig. 15 The top left is the input image extracted from Google Earth, while the top right is the target terrain that river flows over. The middle is a short river scene generated by our method. A larger scene with the same input is demonstrated at the bottom of this figure

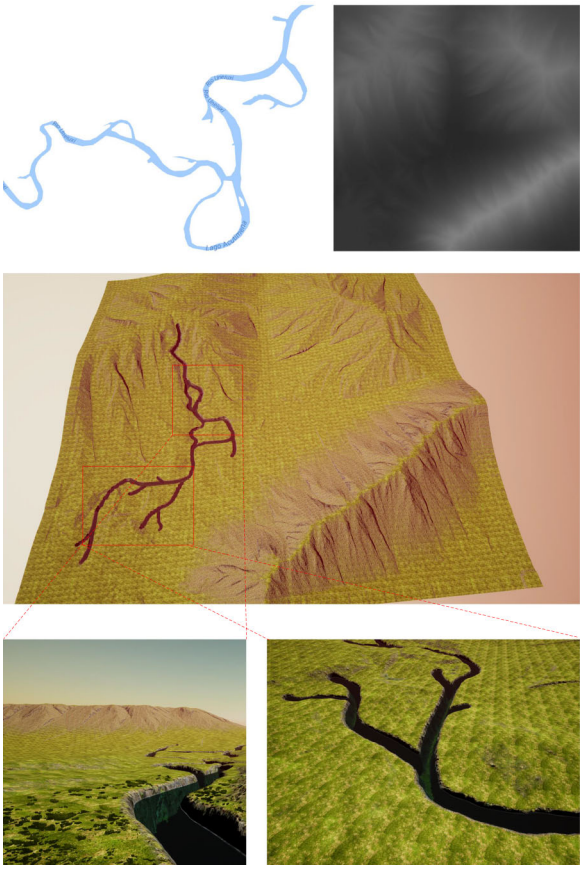


Fig. 16 Another example of the river scene. The input river has an outstanding feature of tortuosity, with distributaries and tributaries. Our system generates a similar river flowing from the mountain to the plain

6.2 Results

The system is implemented in C++, and the computations are performed on a PC with NVidia GeForce GTX850M and Intel Core i7 CPU, running at 2.5GHz with 8 GB RAM. With our system, we are able to calculate several river scenes from different example rivers and terrains. We render the rivers and lakes by using the integrated mesh and for terrains we use texture blending and level-of-detail (LOD) technique. Fresnel reflection and refraction technique is also used for the water rendering. All 3D scenes are rendered by Unreal Engine [12] in real time.

In Fig. 6, we show three different river scenes generated by our procedural model, with user-defined terrain and parameters. The figure shows that our method catches features such as tortuosity, distributaries and tributaries, and can produce various results from appropriate parameters, while maintains the coherence between terrains and rivers. Figures 15 and 16 showcase the river scenes resembling the rivers shown in corresponding images. With provided river image, the interaction process generally takes less than 1 min (Fig. 7). For Fig. 15, the input image shows a river with several short distributaries, and our method creates two rivers with different scale flowing along the terrain from the user-specified starting position to the end. Figure 16 demonstrates another result of our method. With features similar to the input image, our method generates a new river following on the rough terrain. In Fig. 17, various rivers are generated with a same input exemplar and with different scale. These rivers are connected into the single large river while maintaining the consistency between terrains and rivers.

Statistics The overall time consumption of our method is mainly composed of four parts: parameter extraction, initial river network generation, river network optimization, and generation of the 3D river scene. It is necessary to mention that the acceptance condition of river network generation impacts a lot for the time consumption of our method. Also, our algorithms' behavior is affected by random seed since we use a stochastic method. The values of parameters have an obvious impact on the computation time. We set the threshold

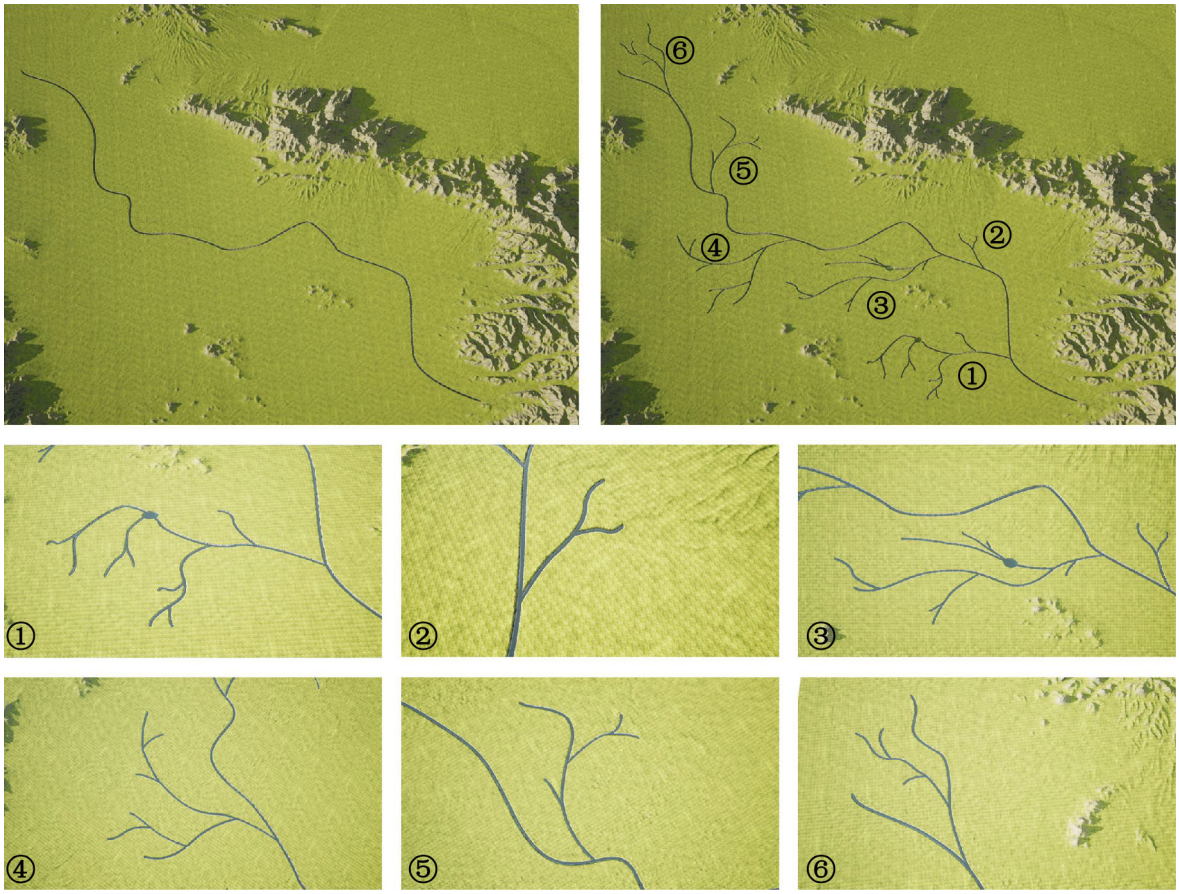


Fig. 17 Taking the first river network in Fig. 6 as the exemplar scene, we added six new rivers connected into an existing long river

Table 5 Computation time for our simulations

	Length (m)	Computation time in second			
		Extraction	Generation	Optimization	3D scene
Figure 6a	8611	–	4.72	–	4.17
Figure 6b	6379	–	4.53	–	4.92
Figure 6c	5068	–	3.17	–	3.47
Figure 15-S	971	0.17	1.12	5.91	2.13
Figure 15-L	6783	0.17	5.03	16.73	3.86
Figure 16	4022	0.21	3.09	8.84	3.69

of energy value (5.3) $\sigma = 0.15$ and count the time consumption at each component.

Examine Table 5, from left to right, the columns of the table list the figure number, the length of result river and the computation time at each algorithm part (including feature extraction, generation of initial river network, optimization of river network, and 3D scene generation). In Table 5, 15-S, -L refer to the smaller and the larger scenes in Fig. 15, respectively. Unlike the classical practice for Reversible jump Markov chain Monte Carlo, our optimization method does not start with random sample. With an initialization similar to the expected result and perturbations in a contiguous space, our method has a rapid convergence (shown in Fig. 10).

Limitations The first limitation is that our method does not guarantee global hydraulic correctness because of the river network only adapting terrain locally. In addition, the proposed approach currently cannot extract the cross-sectional features from an input image. Besides, since the input of our method is a river image and our system generates new river with no interaction, the controllability of the output is not remarkable. However, in this way our system is friendly to the novice user, they just need to provide the photo of river network and our system automatically creates new river adap-

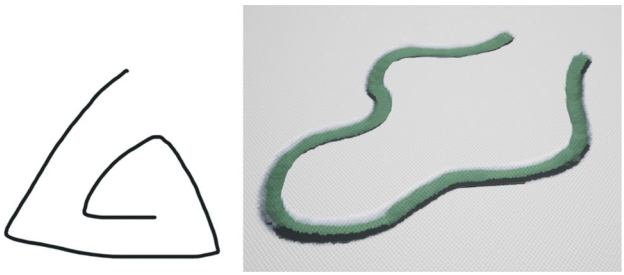


Fig. 18 Our method fails to produce a satisfactory result (right) with a user sketch (left) as the first input. Sketches often contain unnatural or artistic features which cannot be captured by our method

tive to the terrain. Moreover, our method currently handles terrain by using a grid-based approach which brings the memory limitation problem, so that the generated scene is not so huge. In the future, we want to produce river scene from an image at a huge scale while keeping the similarity with the input image. Last, our method works well with river scenes from nature, but we cannot handle scenes with artificial features such as artwork or deliberate sketching (Fig. 18).

7 Conclusion and future work

This paper has introduced a new method for procedural modeling toward river scene creation, with optimized results based on examples. We started with a novel parametric model for river scene generation and expanded the synthesized method to handle the river network, while retaining the consistency between the river and the complicated terrain. Our model could generate various results of river scenery by offering the advantage of compact parameter selection.

To further simplify the user interaction of choosing appropriate parameters, we relied on a natural river image to extract parameters toward a rapid production of river scenes. In order to better satisfy users' demand and expectation, we designed a method to compare two river scenes and introduced an iterative optimization of river scene by using the simulated annealing algorithm. Several results demonstrated a suite of functionalities of our method.

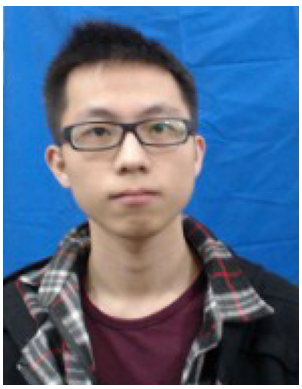
Future work could focus on the global consistency between rivers, terrains and other natural landscape entities such as bog, swamp. The interaction of river segmentation could be improved, using a fully automatic method instead. The input image right now only accommodates the orthographic view of the river scene from the top, and we wish to lift this restriction by allowing images taken from various angles in the near future, while permitting river width to be changed. We also hope to combine our work with physical techniques such as erosion simulation [5], which makes the scene more exquisite.

Acknowledgements This paper is partially supported by Natural Science Foundation of China (No.61532002, 61672237), Natural Science Foundation Grant NSF IIS-1715985, and National High-tech R&D Program of China (863 Program) under Grant 2015AA016404.

References

1. Aliaga, D.G., Vanegas, C.A., Benes, B.: Interactive example-based urban layout synthesis. In: ACM Transactions on Graphics (TOG), vol. 27, p. 160. ACM (2008)
2. Argudo, O., Andujar, C., Chica, A., Guérin, E., Digne, J., Peytavie, A., Galin, E.: Coherent multi-layer landscape synthesis. Vis. Comput. **33**, 1005–1015 (2017)
3. Brandt, J.W., Algazi, V.R.: Continuous skeleton computation by Voronoi diagram. CVGIP Image Underst. **55**(3), 329–338 (1992)
4. Cordonnier, G., Braun, J., Cani, M.P., Benes, B., Galin, E., Peytavie, A., Guérin, E.: Large scale terrain generation from tectonic uplift and fluvial erosion. Comput. Graph. Forum. **35**, 165–175 (2016)
5. Cordonnier, G., Galin, E., Gain, J., Benes, B., Guérin, E., Peytavie, A., Cani, M.P.: Authoring landscapes by combining ecosystem and terrain erosion simulation. ACM Trans. Graph. **36**(4) (2017). https://doi.org/10.1145/3072959.3073667
6. Derzapf, E., Ganster, B., Guthe, M., Klein, R.: River networks for instant procedural planets. Comput. Graph. Forum. **30**, 2031–2040 (2011)
7. Emilien, A., Bernhardt, A., Peytavie, A., Cani, M.P., Galin, E.: Procedural generation of villages on arbitrary terrains. Vis. Comput. **28**(6–8), 809–818 (2012)
8. Emilien, A., Poulin, P., Cani, M.P., Vimont, U.: Interactive procedural modelling of coherent waterfall scenes. Comput. Graph. Forum. **34**, 22–35 (2015)
9. Emilien, A., Vimont, U., Cani, M.P., Poulin, P., Benes, B.: Worldbrush: interactive example-based synthesis of procedural virtual worlds. ACM Trans. Graph. **34**(4), 106 (2015)
10. Fisher, M., Ritchie, D., Savva, M., Funkhouser, T., Hanrahan, P.: Example-based synthesis of 3d object arrangements. ACM Trans. Graph. **31**(6), 135 (2012)
11. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Trans. Graph. **23**, 652–663 (2004)
12. Games, E.: Unreal engine. https://www.unrealengine.com (2007). Accessed 13 Nov 2017
13. Génevaux, J.D., Galin, É., Guérin, E., Peytavie, A., Benes, B.: Terrain generation using procedural models based on hydrology. ACM Trans. Graph. **32**(4), 143 (2013)
14. Génevaux, J.D., Galin, E., Peytavie, A., Guérin, E., Briquet, C., Grosbellet, F., Benes, B.: Terrain modelling from feature primitives. Comput. Graph. Forum. **34**, 198–210 (2015)
15. Guerrero, P., Jeschke, S., Wimmer, M., Wonka, P.: Learning shape placements by example. ACM Trans. Graph. **34**(4), 108 (2015)
16. Han, J., Zhou, K., Wei, L.Y., Gong, M., Bao, H., Zhang, X., Guo, B.: Fast example-based surface texture synthesis via discrete optimization. Vis. Comput. **22**(9–11), 918–925 (2006)
17. Hnaidi, H., Guérin, E., Akkouche, S., Peytavie, A., Galin, E.: Feature based terrain generation using diffusion equation. Comput. Graph. Forum. **29**, 2179–2186 (2010)
18. Horton, R.E.: Erosional development of streams and their drainage basins; hydrophysical approach to quantitative morphology. Geol. Soc. Am. Bull. **56**(3), 275–370 (1945)
19. Hou, F., Qin, H., Qi, Y.: Procedure-based component and architecture modeling from a single image. Vis. Comput. **32**(2), 151–166 (2016)

20. Huijser, R., Dobbe, J., Bronsvoort, W.F., Bidarra, R.: Procedural natural systems for game level design. In: 2010 Brazilian Symposium on Games and Digital Entertainment (SBGAMES), pp. 189–198. IEEE (2010)
21. Ijiri, T., Mech, R., Igarashi, T., Miller, G.: An example-based procedural system for element arrangement. Comput. Graph. Forum. **27**, 429–436 (2008)
22. Kelley, A.D., Malin, M.C., Nielson, G.M.: Terrain Simulation Using a Model of Stream Erosion, vol. 22. ACM, New York (1988)
23. Kelly, G., McCabe, H.: A survey of procedural techniques for city generation. ITB J. **14**, 87–130 (2006)
24. Landes, P.E., Galerne, B., Hurtut, T.: A shape-aware model for discrete texture synthesis. Comput. Graph. Forum. **32**, 67–76 (2013)
25. Mei, X., Decaudin, P., Hu, B.G.: Fast hydraulic erosion simulation and visualization on GPU. In: 15th Pacific Conference on Computer Graphics and Applications, 2007. PG'07, pp. 47–56. IEEE (2007)
26. Merrell, P.: Example-based model synthesis. In: Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games, pp. 105–112. ACM (2007)
27. Nishida, G., Garcia-Dorado, I., Aliaga, D.: Example-driven procedural urban roads. Comput. Graph. Forum. **35**, 5–17 (2016)
28. Nishida, G., Garcia-Dorado, I., Aliaga, D.G., Benes, B., Bousseau, A.: Interactive sketching of urban procedural models. ACM Trans. Graph. **35**(4), 130 (2016)
29. Pajarola, R., Gobbetti, E.: Survey of semi-regular multiresolution models for interactive terrain rendering. Vis. Comput. **23**(8), 583–605 (2007)
30. Prusinkiewicz, P., Hammel, M.: A fractal model of mountains and rivers. In: Proceedings of Graphics Interface, vol. 93, pp. 174–180. Canadian Information Processing Society (1993)
31. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23**, 309–314 (2004)
32. Rusnell, B., Mould, D., Eramian, M.: Feature-rich distance-based terrain synthesis. Vis. Comput. **25**(5), 573–579 (2009)
33. Samavati, F., Runions, A.: Interactive 3d content modeling for digital earth. Vis. Comput. **32**(10), 1293–1309 (2016)
34. Smelik, R.M., Tutenel, T., Bidarra, R., Benes, B.: A survey on procedural modelling for virtual worlds. Comput. Graph. Forum. **33**, 31–50 (2014)
35. Smelik, R.M., Tutenel, T., de Kraker, K.J., Bidarra, R.: A declarative approach to procedural modeling of virtual worlds. Comput. Graph. **35**(2), 352–363 (2011)
36. Šťava, O., Beneš, B., Brisbin, M., Křivánek, J.: Interactive terrain modeling using hydraulic erosion. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 201–210. Eurographics Association, Switzerland (2008)
37. Stava, O., Pirk, S., Kratt, J., Chen, B., Měch, R., Deussen, O., Benes, B.: Inverse procedural modelling of trees. Comput. Graph. Forum. **33**, 118–131 (2014)
38. Tan, P., Zeng, G., Wang, J., Kang, S.B., Quan, L.: Image-based tree modeling. ACM Trans. Graph. **26**, 87 (2007)
39. Wang, F., Kang, L., Li, Y.: Sketch-based 3d shape retrieval using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1875–1883 (2015)
40. Xu, K., Stewart, J., Fiume, E.: Constraint-based automatic placement for scene composition. Graph. Interface **2**, 25–34 (2002)
41. Yu, Q., Neyret, F., Bruneton, E., Holzschuch, N.: Scalable real-time animation of rivers. Comput. Graph. Forum. **28**, 239–248 (2009)
42. Zhang, H., Qu, D., Hou, Y., Gao, F., Huang, F.: Synthetic modeling method for large scale terrain based on hydrology. IEEE Access **4**, 6238–6249 (2016)
43. Zhou, H., Sun, J., Turk, G., Rehg, J.M.: Terrain synthesis from digital elevation models. IEEE Trans. Vis. Comput. Graph. **13**(4), 834–848 (2007)



Jian Zhang is currently a Ph.D. student of School of Computer Science and Software Engineering, East China Normal University, China. He received his B.E. degree in software engineering from East China Normal University in 2014. His research interests include procedural modeling and sketch-based modeling.

Chang-bo Wang is a professor of School of Computer Science and Software Engineering, East China Normal University, China. He received his Ph.D. degree at the State Key Laboratory of CAD & CG, Zhejiang University in 2006, and received B.E. degree in 1998 and M.E. degree in civil engineering in 2002, respectively, both from Wuhan University of Technology. His research interests include physically based modeling and rendering, computer animation and realistic image synthesis, information visualization, and others.

Hong Qin is a Full Professor of Computer Science in Department of Computer Science at State University of New York at Stony Brook (Stony Brook University). He received his BS (1986) degree and his MS degree (1989) in Computer Science from Peking University in Beijing, China. He received his Ph.D. (1995) degree in Computer Science from the University of Toronto. His research interests include Computer Graphics, Geometric and Physics-based Modeling, Computer Aided Design, Computer Aided Geometric Design, Computer Animation and Simulation, Virtual Environments and Virtual Engineering, and others.

Yi Chen is currently a graduate student of School of Computer Science and Software Engineering, East China Normal University. He received his B.S. degree in computer science and technology in 2015. His research interests include terrain modeling and virtual reality.

Yan Gao is an associate professor of School of Computer Science and Software Engineering, East China Normal University, China. He received his B.S. (1993) degree in Computer Science from Nanjing University of Aeronautics and Astronautics, and received his M.S. degree (2002) in Computer Science from Wuhan University of Technology, China. He received his Ph.D. (2006) degree in Computer Science from Shanghai Jiao Tong University. His research interests include computer animation and Geometric modeling and others.