A Practical Terrain Generation Method Using Sketch Map and Simple Parameters

Hua Fei YIN†, ††, Student Member and Chang Wen ZHENG†, Nonmember

SUMMARY A procedural terrain generation method is presented in this paper. It uses a user-drawn sketch map, which is a raster image with lines and polygons painted by different colors to represent sketches of different terrain features, as input to control the placement of terrain features. Some simple parameters which can be easily understood and adjusted by users are used to control the generation process. To further automatically generate terrains, a mechanism that automatically generates sketches is also put forward. The method is implemented in a PC, and experiments show that terrains are generated efficiently. This method provides users a controllable way to generate terrains.

key words: automatic generation, procedural methods, terrain sketch, terrain generation

1. Introduction

Three-Dimensional (3D) modeling is an important research area in computer graphics. In the 3D modeling area, terrain synthesis plays an important role in many applications. Terrain synthesis involves the creation of a model that approximates a terrain resulting from erosional and geological processes. Synthetic terrains are widely used in flight simulation, computer games, movie making, military simulation training, and so on [1], [2].

Terrain models can be created by hand, or automatically by terrain synthesis algorithms. Existing procedural terrain synthesis methods have several shortcomings: their controls supplied to the user are limited or complex, they produce homogenous results, they require users to have professional modeling knowledge and skills, and they are time-consuming. These usually preclude a procedural method from being used in mainstream terrain modeling, and limit the transformation of manual terrain modeling to automation.

In this paper, we propose a terrain synthesis method using a sketch map as input. Users can control the placement of terrain features in an intuitive way, and some adjustable parameters are used to control the final synthesis results of terrain features.

The contribution is that a novel simple way is supplied to generate terrain which is easy to master, controllable, and fast enough.

2. Related Work

Early procedural terrain synthesis methods often related to fractal technologies [3]–[6]. They usually transform frequency of fractional Brownian motion to height values by using Fast Fourier Transform. These methods can generate large-scale terrains in a quite rapid speed. The fault is that they lack intuitive and effective control on the synthesized terrain.

There are also approaches generate terrains by simulating real world physical phenomena (like erosion by wind or water) [7]–[9]. These methods can generate highly realistic terrains, if the user has a deep understanding of the physical laws to implement. But they will consume an extreme long time to generate terrains.

Some researchers generate terrains by making use of image processing and texture synthesis techniques. Szeliski et al [10] generate realistic and controllable terrains by combining variational splines and stochastic fractals together; the method gets other elevation values from existing sparse elevation data by spline interpolation. Zhou et al [11] use nonparametric texture synthesis algorithms to synthesize terrains. It picks appropriate patches from input example height fields, and puts them into the synthesized terrains; the synthesis process is guided by a user-sketched feature map. Rusnell et al [12] generate height fields by least-cost paths in a weighted graph from a set of points with known elevation values. Pang et al [13] proposed an algorithm to synthesize large-scale terrains from a set of images. Terrain blocks are created from the images, and local neighboring blocks are blended together. Chiang et al [14] synthesize terrains using real terrain elevation data. Real terrain units are picked up from database according to user-specified terrain microscopic features, and a minimal cut approach is exploited to stitch adjacent terrain units together to synthesize macroscopic terrain. Brosz et al [15] extract high-resolution details from existing terrains and applied them to increase the resolution of a low-resolution terrain. They all require an initial set of terrains or images which must be obtained from other process. Furthermore, the result is limited by the input and new features cannot be generated.

There are also some other terrain generation methods. Kamal and Uddin [16] proposed an approach towards parametrically controlled terrain generation. Their method is limited only to generate a landscape having one primary mountain peak, and the location of the peak cannot be con-
controlled by the user. Gain et al [17] introduced a sketching interface to procedural terrain generation. Users draw the silhouette, spine and bounding curves of landforms, then terrain is interactively generated to match the sketched constraints using multisresolution surface deformation. Doran and Parberry [18] use intelligent agents to generate unique terrain elevation heightmaps that have specific features. It is much like our method that using automatically generated sketch map as input, and yet our method can use a sketch drawn by users to exactly place terrain features while their method cannot.

The method presented in this paper is intended to generate terrain by procedures with features that novice users can master it quickly, and they can control positions of terrain features exactly in the final result, also the generation time should quick enough. The current methods cannot satisfy all these aspects.

3. Terrain Generation Method

The proposed novel method is controllable, easy to learn, and fast enough to generate terrain. Users can exactly place terrain features by drawing the sketch map, meanwhile, some parameters are used to control the synthesis result, and they can easily be understood by users to make results as expected.

A sketch map drawn by the user is used as input in our method, and the generation process is controlled by some simple parameters which can be adjusted by the user. It has three steps: 1) Initialization: scan the sketch map and classify feature points; 2) Generate terrain skeleton: generate height values for the feature points according to control parameters; 3) Extension: extend and improve terrain features using corresponding parameters.

3.1 Definitions

The terrain features involved are defined as following:

**Plain:** In geography, a plain is land with relatively low relief, that is flat or gently rolling. In our research, to simplify the problem, we define a plain as land with a common height value relatively low in the terrain. Plain feature is the base feature in our generated terrain.

**Mountain:** A mountain is a large landform that rising abruptly from the surrounding land. It refers to a region where there are many such features in our research.

**Canyon:** A canyon is a deep ravine between cliffs often carved from the landscape by a river.

**River:** A large natural stream of water flowing in a channel to the sea, a lake, or another such stream. In this paper, we focus on the generation of river bed which has a lower height than plain.

**Lake:** A lake is a large area of relatively still water surrounded by land. The height of a lake is lower than plain in our research.

3.2 Initialization

In this step, the sketch map \( S(m \times n) \) is read and scanned. The sketch map is a two dimensional raster image with lines and polygons painted by different colors and represent sketches of different terrain features. Each pixel in the image is iterated and the RGB value is got and compared with some predefined values. If it equals to a certain value, a value which indicates terrain feature type is set to the corresponding position in a matrix \( M \) whose dimension is the same as \( S \). Then matrix \( M \) is served as a lookup table from which the feature type of a position can be quickly acquired. A height field \( G \) with the same size is created and all height values in it are initialized to unknown.

3.3 Generate Terrain Skeleton

Every point in height field \( G \) is iterated and its height value is generated according to the value in \( M \). This course can be described by Algorithm 1.

**Algorithm 1** Generate Terrain Skeleton

| Input: indication matrix \( M(m \times n) \), control parameters \( P \) |
| Output: height field \( G(m \times n) \) |
|-----------------------------------------------|
| 01 foreach \( M_{ij} \) in \( M \) |
| 02 switch \( M_{ij} \) |
| 03 case MOUNTAIN: |
| 04 \text{variation} \leftarrow \text{rand}(-\text{mtHghtVar}, \text{mtHghtVar}); |
| 05 \( G_{ij} \leftarrow \text{mtBaseHeight} + \text{variation}; \) |
| 06 break; |
| 07 case CANYON: |
| 08 \text{variation} \leftarrow \text{rand}(-\text{cyHghtVar}, \text{cyHghtVar}); |
| 09 \( G_{ij} \leftarrow \text{cyBaseHeight} + \text{variation}; \) |
| 10 break; |
| 11 case RIVER: |
| 12 \( G_{ij} \leftarrow \text{riverBedHeight}; \) |
| 13 break; |
| 14 case LAKE: |
| 15 \( G_{ij} \leftarrow \text{lakeBottomHeight}; \) |
| 16 break; |
| 17 case PLAIN: |
| 18 default: |
| 19 \( G_{ij} \leftarrow \text{plainHeight}; \) |
| 20 break; |

If it is a mountain or canyon point, generate a base value and a random value according to control parameters \( P \), and the sum of these two values is set as height value of current point; if it is a river, lake or plain point, directly set the height value of current point as a corresponding value in \( P \).

3.4 Extension

Some terrain features are extended as required in this step.
At present we only extend ridges and canyons generated in the previous step to generate full mountains and canyons.

For each point in the height field, generate height values for certain neighboring points indicated by a line segment whose direction is the same as the aspect of this point. This process is: choose the extension direction first; as height values of ridge points are already calculated, the width direction of a mountain or canyon can be decided by the aspect of current point; then points to be covered in this direction are picked up and height values are generated for them.

1) Calculation of Aspects: There are quite a few approximate methods for calculate aspect through elevation grids, we use Horn algorithm [19] here to calculate aspects. It uses eight neighboring pixels for estimation; formulas for calculate aspect through elevation grids, we use Horn algorithm [19] here to calculate aspects.

\[ A = \arctan\left(\frac{n_y}{n_x}\right), \]  
\[ n_x = (e_1 + 2e_4 + e_6) - (e_3 + 2e_5 + e_8), \]  
\[ n_y = (e_6 + 2e_7 + e_8) - (e_1 + 2e_2 + e_3), \]

where \( e_i \) is the height value of a neighboring point.

2) Pick up Height Field Points to be Covered: As the aspect has been calculated, a certain distance from current point to both two sides is searched indicated by the direction to form a line segment, the height field points in this line segment whose features are not RIVER or LAKE are the points to be picked up (the point set is denoted as \( N \)).

This process is much similar to draw lines on grids, so line drawing algorithms can be used. We adopt Bresenham algorithm here; it only uses addition and subtraction of integers, so it is efficient. The inputs of original Bresenham algorithm are coordinates of the two line segment endpoints. But in the extension, we only know the coordinate of the start point (a point in the ridge or canyon line), the obliquity (aspect) and length of the line segment. We can calculate the coordinate of end point \((x, y)\) according to the known data using

\[
\begin{aligned}
x &= \lfloor \text{halfDist} \cdot \sin(\text{aspect}) \rfloor, \\
y &= \lfloor \text{halfDist} \cdot \cos(\text{aspect}) \rfloor,
\end{aligned}
\]

where \( \text{halfDist} \) stands for the half value of the width to be extended, the calculate formula is

\[
\text{halfDist} = \frac{\text{random} \times (\text{mtMinWidth} + \text{mtMaxWidth})}{2 \times \text{space}} + \text{randNum},
\]

where \( \text{mtMinWidth}, \text{mtMaxWidth} \in P \), stand for the mountain or canyon width range to be extended, \( \text{space} \) is the spacing of height field \( G \), \( \text{randNum} \) is a random number.

The line point whose feature is not RIVER or LAKE is picked up to \( N \).

This process can be indicated by Fig. 2. Filled grids stand for mountain feature points, and arrows represent directions to do extension. The aspect of point \( O \) is \( \pi/4 \), so the extension direction is both sides of point \( O \) as showed by the arrow. Points \( A \) and \( B \) are decided using Eq. (4), \( OA \) and \( OB \) equal to \( \text{halfDist} \). Points compose the two line segments are decided by using Bresenham algorithm.

3) Generate Height Values: When the point set \( N \) is decided, generate height values for all points in it; one dimensional random midpoint displacement algorithm is used. Figure 3 shows profiles of a mountain and canyon. Point \( A \) and \( C \) \((A, C \in N)\) are midpoints of line segment \( O \) and \( AE, \) \( EF \) (already known) is the height value of a point in the ridge or canyon to do the extension, \( AB \) is the height value of \( A \), it is calculated by

\[
AB = \frac{EF}{2} + \text{rand},
\]

and \( CD \) is calculated by

\[
CD = \frac{(EF + AB)}{2} + \text{rand},
\]

\( \text{rand} \) is a random generated number. We recursively calculate height values of points in set \( N \).

4. Automatic Sketch Map Generation

In some situations, the user does not care about the exact positions of terrain features, but wants to generate terrains in a simpler and rapid way. So we supply a method for automatically generate the sketch map; combined with generation method discussed in previous section, users can directly generate terrains without drawing a sketch map by hand.

The method generates terrain features in the sketch map in turn, and a set of parameters are used to control the generation process. First, a sketch map is generated with all pixel points initialized to plain features; then lake features are generated, and mountain features are generated at last.

4.1 Generate Sketch of Lake Features

Lake features in the sketch map are points that covered by
a lake. A 3 × 3 window is used to randomly walk around in the sketch map, and values of points covered by the window are set to be lake features. The generation can be described by Algorithm 2.

**Algorithm 2 Generate Sketch of Lake Features**

Input: initialized sketch map $S_{(m \times n)}$, parameters $P'$
Output: new sketch map $S_{(m \times n)}$
01 $i \leftarrow \text{rand}(1, m), j \leftarrow \text{rand}(1, n)$;
02 $\text{lakeArea} \leftarrow \text{rand}(\text{minLakePt}, \text{maxLakePt})$;
03 $\text{currentArea} \leftarrow 0$;
04 while ($\text{currentArea} < \text{lakeArea}$)
05 \quad $S_{i, j} \leftarrow \text{LAKE\_COLOR}$;
06 \quad $\text{currentArea} \leftarrow \text{currentArea} + \text{1}$;
07 \quad foreach point $(p, q)$ in 8 neighboring points of $(i, j)$
08 \quad \quad if $(1 \leq p \leq m \&\& 1 \leq q \leq n)$
09 \quad \quad \quad $S_{p, q} \leftarrow \text{LAKE\_COLOR}$;
10 \quad \quad endif
11 \quad endwhile
12 $(\text{minLakePt}, \text{maxLakePt} \in P')$

Algorithm 2 generates features in one lake; it is executed for several times as indicated by the lake number $(\text{lakeNum} \in P')$ to be generated.

The generation of foothills is much like ridges, but their start points and directions (perpendicular to the ridge direction) are already known, and their lengths and direction change periods are decided by corresponding parameters of the ridge.

**5. Experiment Results**

We implemented our methods in a personal computer (Intel Core 2 Duo E6320 CPU, 2.00 GB RAM, ATI Radeon X1600 Pro graphic card), under Windows XP operating system and based on Visual C++ developing environment.

Figure 4 is a canyon synthesis result from a sketch map generated by extracting canyon axes from a DEM file of Grand Canyon. Figure 5 is mountains generated from a sketch map drawn according to a photo of real terrain, and Fig. 6 is the shaded result. Figure 7 is a generation result containing mountain, river and lake features. Figure 8 shows a terrain generated by using method of automatic sketch map generation; guided by parameters, the sketch map was automatically generated with four lakes and thirty mountains, and the generation time is 0.516 s. Figure 9 shows more rendered results in close-up views. A summary of related statistical data is shown in Table 1.

The final images of result terrains except Fig. 4 were rendered with Terragen (www.planetside.co.uk/terragen), and Fig. 4 (c) was created by using artificial color derived from elevation value. The size of each image and the rendering time are shown in Table 2.

**5.1 Evaluation**

We consider the following aspects to evaluate the results:

**Realism.** From experiment showed by Fig. 5, we can see this method has the ability to synthesize terrain much like the real terrain. But this ability is limited when producing complex terrains. For example, the method can hardly generate a terrain with various types of mountains. The cause is that it uses common control parameters for on feature type.

**Feature Type.** The method can generate common terrain features including mountain, canyon, river, lake and plain. Experiments are showed by Fig. 4 and Fig. 7. It is
able to generate terrain for the general use.

**Usability.** Because users can draw sketch map to intuitively control terrain feature positions, and control parameters are simple, novice users can easily use this method to produce ideal results. Experiments listed above have showed this advantage. They were all carried out by users after only few attempts, and the total time of sketch drawing and parameter setting are all within 10 minutes. To further evaluate the usability user testing is carried out. Users were asked to evaluate the usability of this method by giving a mark between 1 and 5. The result average score is 4.1 points which shows that users think its usability is fine.

**Quality.** To evaluate the quality of the generation results, user testing is also carried out the same as usability
evaluation. The average scores are shown in the last column of Table 2. Though considerable overhead time is required to render images by Terragen software, it indeed helps users understand the result terrain better. As we can find that quality of Fig. 5 (c), Fig. 7 (c), and Fig. 8 (c) is obvious higher than Fig. 4 (c) according to the score. The rendering time consumed by Terragen is affected mainly by the complexity and water area appeared in the view. It will take much longer time in rendering water, so that is why the rendering time of Fig. 7 (c) is much longer than the other two while they are all with the same size.

5.2 Discussion of Automatic Sketch Map Generation

Currently, the automatic sketch map generation method only generates plain, mountain, and lake. So there are two features—canyon and river to be added in future. When adding these two features, most attentions should be paid on maintaining the consistency and realism. For example, a river usually flows from high elevation to low elevation, and it often meanders when passing through a relatively flat area in its lower course.

The result using automatic sketch map generation is showed by Fig. 8. Comparing Fig. 8 (a) and Fig. 7 (a), we can find that the mountain sketch of the former curves not as nature as the later, and lake points in the former distribute in a much wider range. The advantage of this generation method is that it can generate far more feature points than hand drawing in a very short time. Referred to Table 1, the automatic method generates 32,763 mountain points and 371,437 lake points in sketch map in 0.516s, while draw 4,267 mountain points and 242,235 lake points in Fig. 7 (a) by hand may take at least 30s.

| Table 2 | Rendering time and quality evaluation of each image. |
|---------|----------------------------------------------------|
| image   | size      | rendering time | quality |
| Figure 4 (c) | 1200 × 1200 | 2.562s         | 3.1     |
| Figure 5 (c) | 1024 × 768  | 50.750s        | 4.2     |
| Figure 7 (c) | 1024 × 768  | 155.016s       | 3.9     |
| Figure 8 (c) | 1024 × 768  | 69.546s        | 4.1     |

Fig. 6  Shaded terrain.

Fig. 7  A generation result containing mountain, river and lake features.
5.3 Comparison

From Table 1, we can see the height field generation time is short, with no more than 3 seconds in our experiments. In a $512 \times 512$ resolution terrain generation, our method takes about 0.3 seconds comparing to 7~38 seconds of Runsell et al’s method [12]; and in a $1200 \times 1200$ resolution, our method takes about 1.4 seconds, while Zhou et al’s method [11] takes at least 300 seconds to generate the terrain.

Figure 10 shows terrains synthesized by Zhou et al’s method [11]. The result terrain is joined with patches from example terrains, so unexpected terrain features will be introduced, as we can find a short mountain range appears unexpectedly in the bottom left area inside the circle in the left terrain. In their method, similar blocks in the sketch map lead to the selected proper patches from example terrains are also much the same; limited input examples magnify this fault, so we can see that the right terrain is homogenous. Meanwhile, height values in the synthesized terrain are decided by the selected terrain patches, the user cannot control them efficiently, and the result is limited by the input, so new features cannot be generated.

Figure 11 shows a comparison of our method with Runsell et al’s method [12]. In their method, terrain features are associated with profiles, so slopes of same features will be the same; this leads to the similarities on slopes in the generated terrain (see (b)), and the terrain looks over smoothly, as shown in (d).

Smelik et al [2] generate terrain by a user-specified 2D raster image. The final terrain is integrated by several terrain layers which are separately edited in advance. Their framework contains more feature types, but only earth layer and vegetation layer generation are realized, so our method contains more supported feature types. To generate mountain feature, users should draw a sketch much like a 2D contour map in their method, so the control will be more precise than
our method, while the drawing time consumed is obviously longer than ours. Because the frame of their method aims to including several terrain layers, the consistency management of interacting features is a big challenge, yet they do not discuss in the paper.

Gain et al[17] generate terrain by drawing sketches of different feature types in 3D space. Because it has to draw several types of curve, the drawing of this method is more complex than our method, while its controllability on terrain features is higher than ours. Their method can import and modify existing terrains and ours just generates totally new terrains, so their method is more flexible. It takes at least 2.3s to generate a 512 × 512 resolution terrain with their method, comparing 0.328s with our method.

6. Conclusions and Future Work

A procedural terrain generation method is described in this paper. It is easy to learn, controllable, and fast which related methods can not possess all these features. By using a user-drawn sketch map and some simple parameters, the user can generate terrains according to their intentions conveniently. An automatic sketch map generation method is also put forward, so that users can skip the step of drawing a sketch map by hand and directly generate a terrain controlled by certain parameters. This method can generate terrains at a rapid speed, and novice users can easily and intuitively add just the control parameters; this supplies a feasible way for the transition from manual to automated terrain modeling.

Because synthesized by common parameters, though some random numbers are used, same terrain features are still showing certain similarities in our method. So a direction of the future work is to further divide and extend terrain features to enrich synthesized results. The generation of river and lake features is still coarse, so another direction is to improve them, and add canyons and rivers in the sketch map automatic generation.

Acknowledgments

This work has been supported by the National High Technology Research and Development Program (“863” Program) of China (Project No. 2009AA01Z303). The authors would like to thank the anonymous reviewers for their valuable comments and advices.

References

[1] M. Nitsche, C. Ashmore, W. Hankinson, R. Fitzpatrick, J. Kelly, and K. Margenau, “Designing procedural game spaces: A case study,” Proc. FuturePlay, pp.10–12, London, Britain, 2006.
[2] R.M. Smelik, T. Tutenel, K.J. de Kraker, and R. Bidarra, “Declarative terrain modeling for military training games,” Int. J. Comput. Games Technol., vol.2010, Article no.2, Jan. 2010.
[3] A. Fournier, D. Fussell, and L. Carpenter, “Computer rendering of stochastic models,” Commun. ACM, vol.25, no.6, pp.371–384, June 1982.
[4] G.S.P. Miller, “The definition and rendering of terrain maps,” Proc. 13th Ann. Conf. Computer Graphics and Interactive Techniques, pp.39–48, Dallas, USA, 1986.
[5] J.P. Lewis, “Generalized stochastic subdivision,” ACM Trans. Graphics, vol.6, no.3, pp.167–190, July 1987.
[6] S. Stachniak and W. Stuerzlinger, “An algorithm for automatic fractal terrain deformation,” Proc. 8th Int. Conf. Computer Graphics and Artificial Intelligence, pp.64–76, Limoges, France, 2005.
[7] A.D. Kelley, M.C. Malin, and G.M. Nielson, “Terrain simulation using a model of stream erosion,” Proc. 15th Ann. Conf. Computer Graphics and Interactive Techniques, pp.263–268, Atlanta, USA, 1988.
[8] F.K. Musgrave, C.E. Kolb, and R.S. Mace, “The synthesis and rendering of eroded fractal terrains,” Proc. 16th Ann. Conf. Computer Graphics and Interactive Techniques, pp.41–50, Boston, USA, 1989.
[9] K. Nagashima, “Computer generation of eroded valley and mountain terrains,” Vis. Comput., vol.13, no.9–10, pp.456–464, 1997.
[10] R. Szeliski and D. Terzopoulos, “From splines to fractals,” Proc. 16th Ann. Conf. Computer Graphics and Interactive Techniques, pp.51–60, Boston, USA, 1989.
[11] M. Zou, J. Sun, G. Turk, and J.M. Rehg, “Terrain synthesis from digital elevation models,” IEEE Trans. Vis. Comput. Graph., vol.13, no.4, pp.834–848, July/Aug. 2007.
[12] B. Runsell, D. Mould, and M. Eramian, “Feature-rich distance-based terrain synthesis,” Vis. Comput., vol.25, no.5–7, pp.573–579, 2009.
[13] M. Pang and R. Zhao, “Algorithm for synthesizing large-scale virtual terrain from images using radially weighted blending,” J. Nanjing University of Science and Technology (Natural Science), vol.33, no.4, pp.438–443, Aug. 2009. (in Chinese)
[14] M.Y. Chiang, S.C. Tu, J.Y. Huang, W.K. Tai, C.D. Liu, and C.C. Chang, “Terrain synthesis: An interactive approach,” Int. Workshop Advanced Image Tech., pp.533–538, Jeju Island, South Korea, 2005.
[15] J. Brosz, F.F. Samavati, and M.C. Sousa, “Terrain synthesis by example,” Proc. Int. Conf. Computer Graphics Theory and Appli-
[16] K.R. Kamal and M.Y.S. Uddin, “Parametrically controlled terrain generation,” Proc. 5th Int. Conf. Comput. Graph. Interactive Tech. Australia and Southeast Asia, pp.17–24, Perth, Australia, 2007.

[17] J. Gain, P. Marais, and W. Strasser, “Terrain sketching,” Proc. 2009 Symp. Interactive 3D Graphics and Games, pp.31–38, Boston, USA, 2009.

[18] J. Doran and I. Parberry, “Controlled procedural terrain generation using software agents,” IEEE Trans. Comput. Intell. AI in Games, vol.2, no.2, pp.111–119, 2010.

[19] B.K.P. Horn, “Hill shading and the reflectance map,” Proc. IEEE, vol.69, no.1, pp.14–47, Jan. 1981.

**Hua Fei Yin** received the B.E. and M.S.E. degree in software engineering from Huazhong University of Science and Technology, Wuhan, in 2007 and 2009. Currently he is working towards the Ph.D. degree in computer application technology at Institute of Software, Chinese Academy of Sciences. His research interests include GIS data processing, procedural content generation, and virtual reality.

**Chang Wen Zheng** received the Ph.D. degree in control science and engineering from Huazhong University of Science and Technology, Wuhan, in 2003. He is currently a professor in Institute of Software, Chinese Academy of Sciences, Beijing. He has more than 40 papers in journals and conferences, and nearly 30 of them are embodied by SCI and EI. His research interests are in information processing and integration, computer simulation, and artificial intelligence.