Feedback model inspired by biological development to hierarchically design complex structure

高木, 英行
Department of Acoustic Design, Kyushu Institute of Design

OHNISHI, Kei
Kyushu Institute of Design

Feedback Model Inspired by Biological Development to Hierarchically Design Complex Structure

Kei OHNISHI* and Hideyuki TAKAGI**

Kyushu Institute of Design: *Graduate School, **Dept. of Art & Information Design

4-9-1, Shiobaru, Minami-ku, Fukuoka 815-8540, Japan

Tel&Fax: +81-92-553-4555, E-mail: {o-kei,takagi}@kyushu-id.ac.jp, URL: http://www.kyushu-id.ac.jp/~takagi/

***Abstract*—**

We propose a new feedback system, the *BiDevelopment System*— inspired by the development mechanism of Drosophila—and discuss possible computer graphics and data compression applications. Drosophila's development forms complex organs from a fertilized egg by hierarchically diffusing different protein densities roughly among cells during the earlier stages and minutely during the later stages. The proposed system models this mechanism and realizes a hierarchical rough-to-detailed or simple-to-complex design, and vice versa. Unlike many conventional feedback systems, it is possible to embed our intended rough design into the mechanism, which is the most significant feature of the proposed system. We compare the BiDevelopment System to biological development with analogies, explain two example algorithms, create complex images from a simple seed code, and discuss its opposite application, data compression.

Keywords: biological development, feedback system, hierarchical design, computer structures, data compression

1 Introduction

Recently, many proposed engineering methods have been inspired by biological behavior and structures, such as evolutionary computation [2], neural networks, artificial immune systems [3], ant colony systems [4, 5], and so on. The reason is that we can expect to obtain useful hints on creating new methods from the biological behavior and structures we do not know well.

Some of these methods are equivalent to the approaches used in artificial life (A-Life). These approaches create macroscopic structures or functions by only using local interaction between their elements. In these approaches, the macroscopic structures or functions are often quantitatively undefined but qualitatively defined. Consequently, we need to adjust the local interaction rules for our purpose. However, the A-Life approaches have several advantages. One advantage is that these approaches have the possibility to realize structures or functions that are barely realized by a centralized control system.

Simple systems using the A-Life approaches are feedback systems such as cellular automata [10] and L-systems [6, 9]. These systems can create complex structures from a simple initial state. Complex structures are created by repeatedly applying the rules of rewriting symbols corresponding to their structures. It is simple but difficult task for us to estimate the final structure from the initial state and rules because the rules are applied locally. Their effective use may be to discover or to create interesting structures by adjusting the rules and the initial state by trial and error. Interesting structures that we could have never imagined may be discovered by chance.

The first purpose of this paper is to propose a new feedback system inspired by biological development, which we call the "BiDevelopment System." The proposed system uses a mechanism that hierarchically designs a structure from simplistic to detailed, to create our roughly intended structures. The second purpose is to investigate the capability of the proposed system to create the various intended rough structures. The third purpose is to discuss the possibility of applying this system to data compression that encodes complex information as the opposite solution to the proposed system.

Artificial development models focusing on the functional growth of infant arms were applied to robotics [7, 8]. Our development model focuses on the cell division which forms a complex body from an egg cell.

Later, we explain biological development mechanism that has greatly inspired our work.

Almost all multicellular living things develop into adults from a fertilized egg. The initial fertilized egg has design information for its development, which is described in genome and is usually correctly copied into other cells when cells are divided. Each cell consequently proceeds its development process using the same design information in the initial fertilized egg. Various kinds of cells organized in an adult body are generated during the development process, and the differences in the cells are caused by the difference of the proteins generated in the cells according to genes on genome.

Animals and specific organs in plants, such as flowers, become specified shapes formed by the development. Although the development mechanisms of most species are still unknown, that of Drosophila, a kind of fly, has been well studied. We were interested in creating various kinds of cells and correctly assigning them in the development process [1].

Various cells are hierarchically generated and simultaneously assigned in the Drosophila's development. The development has two important procedures that (1) provide positional information to each cell and (2) make the cell memorize the effect of the information. The role positional information is to specialize the cell.

The first step of the procedure (1) is to generate different of protein densities in a fertilized egg so that the fertilized egg can become a set of small cells. Each small cell has different protein densities which are the positional information according to that in the initial fertilized egg. In this step, each cell does not obtain the information about its final organ concretely but the approximate positional information such as head and abdominal regions.

The second step of the procedure (1) is to diffuse a new protein with a different density among the cells from specific subsets of cells, which provides more detailed positional information to the cells. The subsets of cells are determined by the past distribution of protein density.

Each cell gradually obtains positional information by repeating the procedure (1). The rough positional information is obtained in the earlier stages and the detail is obtained according to body growth in the later stages. The early stages of this procedure is shown in Figure 1.

The history of the procedure (1) for each cell hierarchically determines the final organ. Cells continue to generate special proteins according to their obtained positional information, which results the procedure (2).

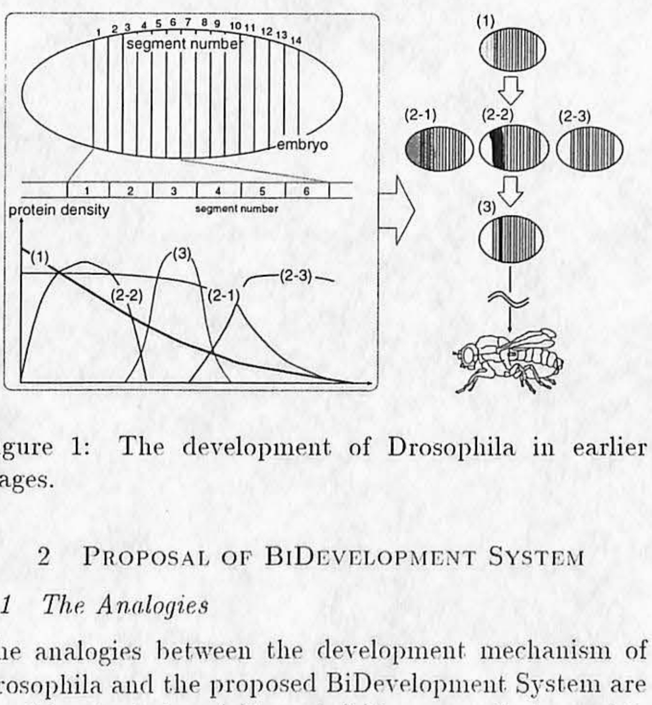

Figure 1: The development of Drosophila in earlier stages.

2 Proposal of BiDevelopment System

2.1 The Analogies

The analogies between the development mechanism of Drosophila and the proposed BiDevelopment System are listed in the below (a)'s and (b)'s, respectively, and is illustrated in Figure 2 (left). The numbers attached in the figure correspond to the following seven analogies.

1. (a) Adult Drosophila consists of various cells. (b) Structure is created by various elements.
2. (a) Only genome in a fertilized egg is a plan and progress table in development. (b) The structure is created according to only a character code.
3. (a) A body is hierarchically formed from simplistic to detailed. The procedure in one hierarchy follows 4(a)→5(a)→6(a). (b) Structure is hierarchically created from simplistic to detailed. The proposed system is realized as a feedback system by corresponding one hierarchy to its main processing parts. The main processing parts follows 4(b)→5(b)→6(b) and the feedback is 7(b).
4. (a) The protein diffused among all or some cells gives them positional information. (b) There is a mechanism to generate positional information among all or some elements.
5. (a) Each cell acts in response to its own positional information, for example, by generating special proteins. (b) Each element acts in response to its own positional information.
6. (a) The state of each cell after acting in response to its positional information is memorized toward the final state. (b) The state of each element after acting in response to its positional information is memorized to form the final output.
7. (a) The protein that provides the current positional information is made from the protein that provided past positional information. Each cell is gradually led to the final organ by the history of the given various positional information. (b) The mechanism that generates the current positional information from past one is prepared. The choices of each element value gradually becomes fewer according to the history of given various positional information, and the system output becomes more detailed.

2.2 The Conceptual Framework

The conceptual framework of the BiDevelopment System is shown in Figure 2 (right). The numbers without parentheses correspond to the analogy number in section 2.1, and those with parentheses correspond to the processing part numbers in section 3. The necessary parameters for every processing part are written in a character code sequence, initialization part, and rules. The character code is a plan and progress table that corresponds to the genome information in biology.

The BiDevelopment System has an initialization part and three major processing parts. The initialization part (0) defines the area where system outputs exist. The processing part (1) generates the global positional information that determines what each element processes. The processing part (2) lets each element be autonomously

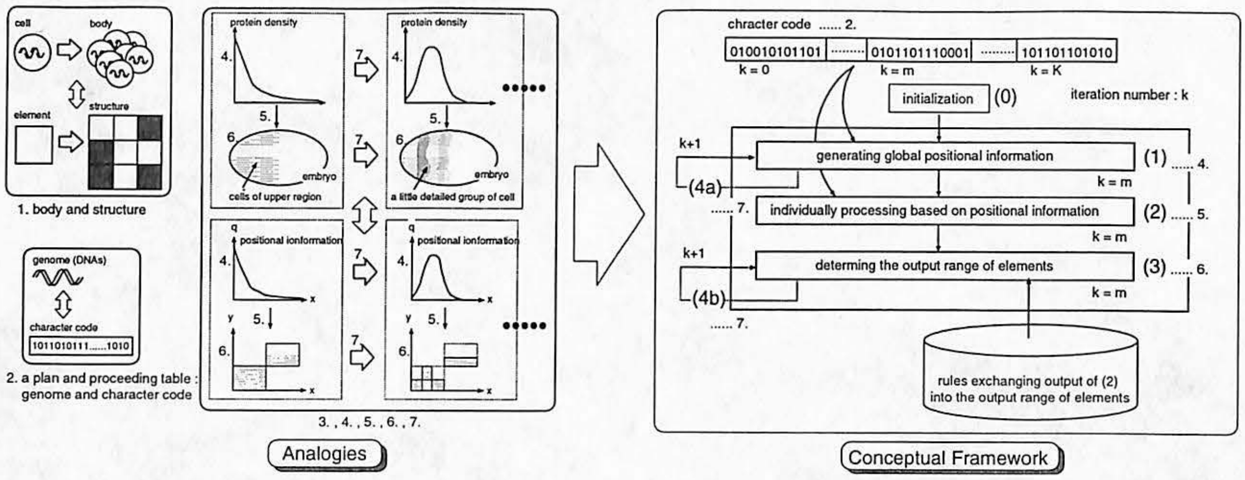

Figure 2: The analogies between the BiDevelopment System and the development mechanism of Drosophila (left). The conceptual framework of the BiDevelopment System (right).

process based on its positional information. Local mutual interaction among elements may take place. The processing part (3) exchanges the output of the processing part (2) into the output range of elements. The final output value of each element is determined by the feedback process (4b).

The part that generates data related with structure is processing part (3). The values that the feedback process at the processing part (3) generates after certain number of feedback loops determine the final structure of the system output. The processing parts (1) and (2) sequentially generate the structure of the final output pattern from the BiDevelopment System.

The processing parts (1) and (3) include feedback processes (4a) and (4b), respectively. The feedback process (4a) generates the positional information of the next state from that of the previous state. The feedback process (4b) gradually narrows the range of output values of each feedback process by determining the final output values.

The advantage of the proposed BiDevelopment System is the embedded rough-design structure and various possible generated structures under the given rough-design structure. Conventional feedback systems, such as the L-System or cellular automaton, apply only one rule to each element at each feedback loop. It is difficult to design an entire final structure from the beginning and embed it into the system. Since each processing part of the proposed system determines final detail from the basic structure, it is easier to embed the rough-design structure before the proposed system runs. The trade-off is that as the number of processing parts increases as mentioned above, its system design becomes a little difficult.

Two devices are implemented in the BiDevelopment System to realize these advantages. The processing part (1) is designed to generate the structure of the final output patterns in order to embed the rough design structure of the designer. The diversity of structures in the detail level is realized by changing the parameters in the character code, the input to each processing part. The combination of these two devices allows the determination of the detail pattern at the processing part (2) under the restriction of the basic structure defined at the processing part (1).

3 Example Algorithms of BiDevelopment System

3.1 *Algorithms flow*

In this section, two algorithms under the framework of the proposed BiDevelopment System are shown and applied to simple tasks to demonstrate its concrete usage. Note that there are various algorithms in the framework of the BiDevelopment System and these two algorithms are just simple examples.

The example algorithms generate y values for given x coordinates, where Algorithm1 is applied in a fixed area $[x_{min}, x_{max}]$, and Algorithm2 gradually expands its applicable area. Figure 3 shows the flow of processing parts in these algorithms. The processing part numbers, (0) – (4b), in the figure correspond to the those of the conceptual framework in section 2.2.

Several parameters used in some processing parts are given in character code and drive the proposed system. Since some of them depend on the given tasks, we first explain toy tasks, then describe the example algorithms in the following sub-sections.

3.2 *Example tasks*

(1) The task for the Algorithm1

The task is to draw a figure in a (x, y) space by generating y values for x coordinates in fixed closed area, $[x_1, x_n]$.

(2) The task for The Algorithm2

The task is to draw a figure in a (x, y) space by generating y values for x coordinates that starts from $[x_{init1}, x_{init2}]$ and expands to $[x_1, x_n]$, where $x_1 \leq x_{init1} \leq x_{init2} \leq x_n$.

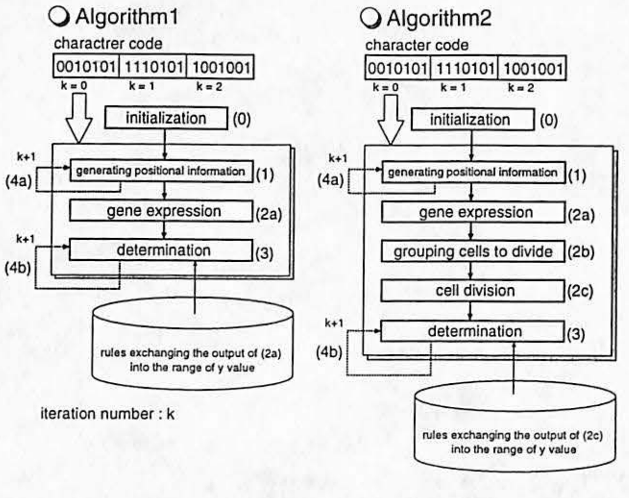

Figure 3: Process flow of Algorithm1 and Algorithm2.

Each part processes these tasks K times including $(K-1)$ feedback processing loops. Suffix k is used in section 3.3 to identify the k-th process at each part. x_1 and x_n are initial input values to the BiDevelopment Systems for the Algorithm1, and x_{init1}, x_{init2}, x_1, and x_n are those for the Algorithm2.

The parameters used for k-th processing at processing parts (1), (2a), (2b), and (2c) are independently described in the character code. This is the main point why and where we can embed our rough design intention into the BiDevelopment System. A designer needs to prepare a rule-base whose rules convert the output of the processing part (2) to the y range being able to take place at the processing part (3). This is the point where a designer can control the way or degree of reducing the y range.

3.3 Processing Parts of the Algorithms

Common parts of Algorithm1 and Algorithm2 are explained together.

(0) initialization

The processing part (0) is basically common in two algorithms and inputs the range of x space where y values are generated to draw a figure. The initial data inputted here are (x_1, x_n) for Algorithm1 and $(x_{init1}, x_{init2}, x_1, x_n)$ for Algorithm2.

(1) generating positional information

The processing part (1) is commonly used in two algorithms and gives positional information of each x coordinate using diffusion functions such as Eq. (1) (see Figure 4 (1).) Let us call the M_k in Eq. (1) a diffusion function from the analogy to diffusing protein among cells in biological development. Any function is available for the diffusing functions, and we adopt a Gaussian function for M_k in this paper. The positional information is determined by the $M(x, A_k, \mu_k, \sigma_k)$ whose parameters, A_k, μ_k, and σ_k are read from the character code.

$$M(x, A_k, \mu_k, \sigma_k) = M_k(x) = A \exp^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}} \qquad (1)$$

Although Algorithm1 and Algorithm2 use only one diffusion function as in Eq. (1), other algorithms may use multiple diffusion functions. Designers can embed their design intentions of the output of the BiDevelopment System and roughly control its shape by controlling the assignment of the diffusion function(s).

(2a) gene expression

The processing part (2a) is commonly used in two algorithms and divides x into several areas according to the s_i read from the character code and $M_k(x)$ values (see Figure 4 (2).) Same labels are given to the divided x areas (see 0 - 3 at the bottom of Figure 4 (2).) We compared the given labels to the genetic information and named this processing part as a *gene expression.* When n diffusion functions are used at the processing part (1) in other algorithms, each x coordinate is given n divided group labels.

(2b) grouping cells to divide

The processing part (2b) is used only in Algorithm2 and divides x coordinates into *copy groups* whose x copies its copy groups label onto another x coordinate based on a copy rule given to each copy groups.

The divided x areas in the processing part (2a) simply and directly become the copy groups in Algorithm2. However, when multiple diffusion functions are used in different algorithms, interpretation rules are need to transform the combination of multiple divided area labels given to each x to one copy group label.

(2c) cell division

This processing part (2c) is only used in Algorithm2 and copies the copy label of each x coordinate onto another x coordinate based on a copy rule. The copy rules are determined by the parameters read from the character code and consist of three values: the copy direction, the amount of copying, and the distance to the copied coordinate. The output of this processing part (2c) is the overlapped pattern of copy group labels at each x coordinate.

(3) determination

The processing part (3) is commonly used in two algorithms and converts the group labels outputted from the previous processing part (2a) or (2c) together with previous y range into the range of y values. The Algorithm1 directly converts the label to y ranges (see Figure 4 (3).) and Algorithm2 converts the overlapped pattern of the copy group labels at each x coordinate to y ranges. The conversion rules that are different in each processing time, k, must be prepared.

Figure 4 shows examples of the first processing result at processing parts (1), (2a), and (3).

(4) feedback

This processing parts (4a) and (4b) that are feedback processing at the processing parts (1) and (3), respectively, are commonly used in two algorithms.

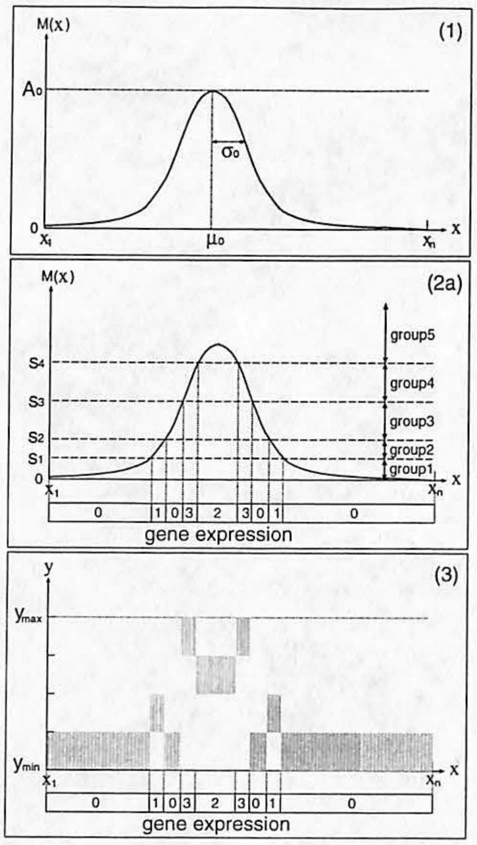

Figure 4: Example outputs of the first processing, $k = 0$, at the processing part (1) generating positional information, the processing part (2a) of gene expression, and the processing part (3) of determination.

(4a) feedback for the processing part (1)

The processing part (4a) calculates μ_k for diffusion function $M_k(x)$ from $M_{k-1}(x)$. It first reads m from the character code, substitutes the m for Eq. (2), and determines μ_k, where μ_0 is given in the character code.

$$\mu_k = \begin{cases} x & if \quad M_{k-1}(x-1) \leq m < M_{k-1}(x) \\ x+1 & if \quad M_{k-1}(x) \leq m < M_{k-1}(x-1), \end{cases} \tag{2}$$

where x takes integer numbers.

When multiple μ_{ki} are obtained, $\sum_i M(x, A_k, \mu_{ki}, \sigma_k)$ is used instead of the Eq. (1) in the processing part (1), and is normalized when its maximum value exceeds 1.0. Other necessary parameters for Eq. (1), A_k and σ_k, are read from the character code.

(4b) feedback for the processing part (3)

According to the feedback processing (4b), the range of y is gradually reduced (see Figure 5). The rules that determine the new ranges of y from the past ranges y and group labels must be previously prepared in a rule-base at the processing part (3). Designers can embed their design intention and roughly control the detail shape of the final system output by changing these rules.

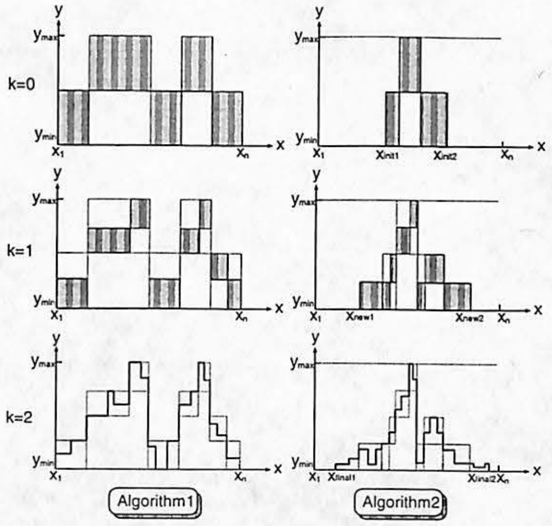

Figure 5: Example outputs of the processing part (3) of Algorithm1 and Algorithm2 according to the feedback processing (4b), where k is the number of this feedback. The dark areas are the ranges of y values that become narrower and in detail according to k.

4 Applications to Images Generation and Discussion on Data Compression

In this section, we demonstrate that the BiDevelopment System creates complex images from a simple character code based on a designer's intended basic design. The algorithm used is similar to the Algorithm2 in section 3 except for a 2-D target space, (x_1, x_2), and the 3 kinds of diffusion functions combining 10 Gaussian functions. As there are three diffusion functions are used, three kinds of division area labels are generated at processing part (2a). Images are displayed by converting the generated y values to the brightness of pixels.

To generate leaf-like images, the intended basic design would be to realize a symmetric and fan-shaped structure. The symmetric structure can be realized by symmetrically assigning diffusion functions with respect to a certain point on x_1 and x_2 axes, respectively, in processing part (1). The fan-shaped expansion can be realized by setting copying directions in copy rules used in processing part (2c) to symmetric with respect to the certain points.

Figure 6 shows three generated images whose initial start area is a circle in the center of each image. They are created by changing the parameters in the character code under same algorithm condition.

Regardless of the artistic view of these images, a simple character code consisting 736 parameters created these complex images according to our embedded design intention of *symmetric* and *fan-shaped*. We may be able to generate further complex images by combining multiple algorithms for the BiDevelopment System. It is also possible application of the BiDevelopment System to generate non-graphic media, such as acoustic signal, based

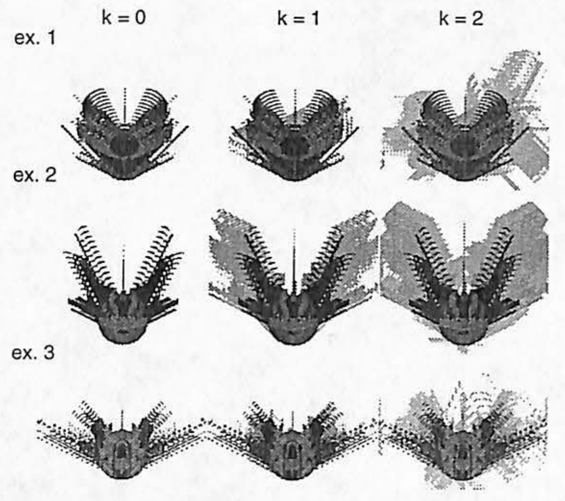

Figure 6: Three example outputs of the BiDevelopment System being embedded *leaf-like* design intention. Images from left to right are the outputs of the processing part (3) and its first and second feedback processing (4b). Since their outputs are not y values but the ranges of y values, which cannot be drawn, except $k = 2$, the minimum value of each y range is displayed in Figure 6 as representative.

on the structures that the proposed system generates.

Data compression is another application of the BiDevelopment System. Although we primarily discussed generating images of complex structure in this paper, data compression is possible by expressing images of complex structure with a simple character code.

Suppose that Figure 6 are target images to be compressed. First, we or analysis tools evaluate the targets, where the outer shapes and brightness distributions are analyzed. We will find out that the images are symmetric and fan-shaped. Even if the brightness distributions cannot be easily analyzed, we can optimize the rules used in processing part (3) to minimize the error between the target images and a synthesized image using optimization techniques. Second, obtained analytical knowledge is embedded in the assignment of diffusion functions in processing part (1), copy rules in processing part (2c), and a rule-base in processing part (3). Finally, these two processes are repeated, and parameters and rules are adjusted until the difference between the synthetic image and the target images becomes less than a certain threshold. The obtained character code becomes the compression result of the target image.

5 Conclusion

We proposed the BiDevelopment System inspired by biological development that generates a complex output from simple codes and rules and extracts simple rules from a complex input. Different from conventional complex systems, it is easy to embed our intended basic design into the proposed system.

We showed two example algorithms in the framework of the proposed BiDevelopment System. We applied them to the toy tasks that generate symmetrically leaf-like images and showed that our design intention was realized in the generated images. It is needed to combine the example algorithms and/or create new algorithms that embed our intended complex designs and generate more complex images such as desks, animals, or vehicles.

We discussed the application of the BiDevelopment System to the data compression, described how to apply the system, and showed the possibility to compile simple structured images to a simple character code. Data compression for more complex structured images are in the next step of the BiDevelopment System research.

As we have already obtained the framework that generates various complex outputs based on our explicit design intent and compress the complex figures to a simple code, it is expected that this tool will be further developed for practical use.

References

[1] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Robert and J. D. Watson, *Molecular biology of the cell,* Garland Publishing, New York, 1989.

[2] T. Bäck and H. P. Schwefel, "Evolutionary computation : An overview," Proc. of The Third IEEE Conference on Evolutionary Computation (ICEC96), Nagoya, Japan, pp.20–29, 1996.

[3] (Eds.) D. Dasgupta, *Artificial immune systems and their applications*, Springer-Verlag, Berlin/Heidelberg, Germany, 1999.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," IEEE Trans. on Systems, Man, and Cybernetics-Part B, vol.26, no.2, pp.29–41, 1996.

[5] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," IEEE Trans. on Evolutionary Computation, vol.1, no.1, pp.53–66, 1997.

[6] A. Lindenmayer, "Mathematical models for cellular interaction in development, PartI and PartII", Journal of Theoretical Biology, vol.18, pp.280–315, 1968.

[7] G. Metta, G. Sandini, and J. Konczak, "A developmental approach to sensori-motor coordination in artificial systems," IEEE Int'l Conf. on System, Man, and Cybernetics (SMC'98), NY, USA, pp.3388–3398 (1998).

[8] G. Metta, G. Sandini, and J. Konczak, "A developmental approach to visually-guided reaching in artificial systems," J. of Neural Networks, vol.12, no.10, pp.1413–1427 (1999).

[9] P. Prusinkiewicz and A. Lindenmayer, *The algorithmic beauty of plants*, Springer-Verlag, New York, 1990.

[10] S. Wolfram, *Cellular automata and complexity: collected papers*, Addison-Wesley, Reading, MA, USA, 1994.