EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2006)

Sketching Reaction-Diffusion Texture

Ly Phan[1] and Cindy Grimm[2]

[1]Southern Arkansas University, Arkansas, United States
[2]Washington University in St. Louis, Missouri, United States

Abstract

In this work, we present an interactive interface for sketching synthesized textures. Reaction-Diffusion (RD) is used as the basis for texture synthesis. RD allows an unlimited amount of non-repeating texture and offers great flexibility for mapping textures to arbitrary surfaces. However, it can be difficult to find starting values of parameters that will produce interesting patterns. We use machine learning to resolve the difficulty of determining appropriate initial values of the RD system. The system described here allows a user to sketch a pattern of spots or stripes with arbitrary orientations, and then automatically generates a pattern with the same attributes as the sketch. It also allows the user to interactively create more complex textures by adding another layer of pattern, as well as manipulate the color of the resulting texture. We also show that this procedure can be applied to realistic 3D surfaces.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture; I.3.8 [Computer Graphics]: Applications

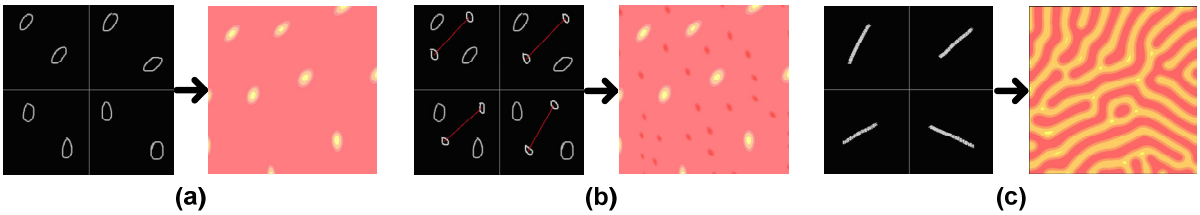


Figure 1. *(a) User sketches ellipses with different orientation. Our system generates dots with similar size, shape, and orientation. (b) User adds smaller ellipses to the sketch. The system generates smaller dots in between larger dots. (c) User sketches lines with different slopes. The system generates random stripes with similar orientation.*

1. Introduction

Textures are two-dimensional images or three-dimensional volumes that can be mapped onto an object's surface. Very often, the object's surface is larger than the texture sample available, hence the need for texture synthesis – generating more of a texture from a given sample. There are two main types of textures: raster textures, and procedural textures. Raster textures can be scanned pictures or painted images. They are ready-to-map, but the cost of memory storage is very high and resampling and mapping them to an arbitrary surface can be very difficult. Procedural textures, on the other hand, are defined mathematically, and therefore take little memory to store. They are also easier to resample and map to a surface. In this paper, we focus on textures generated from a method called Reaction-Diffusion. In addition to allowing an unlimited amount of non-repeating texture, this approach also offers greater flexibility for both controlling the attributes of patterns and mapping textures to arbitrary surfaces.

Reaction-Diffusion (RD) was first introduced as a model of morphogenesis, a biological pattern-formation process in which two or more chemicals – called morphogens – diffuse over a surface and react with each other. The differences in concentration of morphogens form certain animal coat patterns, such as spots and stripes [Tur52]. This process is defined by differential equations, which control the change in concentration of one morphogen relative to other morphogens over time.

A RD system consists of numerous parameters, a few of these being the reaction rates k/s, the diffusion rates D, and the random factor β (Section 3). At present there is no intuitive way to determine the values of these parameters that will produce interesting patterns, much less a pattern that has certain attributes. Thus, although RD can produce interesting and varied textures, it is difficult for a user to use a RD system without having a deep knowledge of its internal workings. We solve this problem by developing an interactive interface that allows users to sketch a rough version of their desired pattern. Our system then automatically generates a texture sample that contains similar features (Figure 1). This is done by analyzing the sketch's attributes and using machine-learning technique to map these attributes to the appropriate starting parameters. Our approach allows for more control over the attributes of the patterns produced by the RD system (in case of Figure 1, the size and orientation of spots and stripes). The system also includes a coloring interface that gives the user freedom to vary the color of the generated textures. Finally, we show that our approach can be applied to generate patterns, as defined by a user, on a 3D surface.

The rest of this paper is organized as follows. In Section 2, we present our interface for sketching RD textures. In Section 3, we provide a brief background of RD systems. In Section 4, we summarize the previous work that has been done on generating textures with RD. Section 5 contains a detailed description of the implementation of this system, including our extension to existing techniques and our application of machine learning to solve the problem of finding the appropriate parameter values. In Section 6, we illustrate how our method extends to 3D surfaces. We conclude with a discussion of the future work in Section 7.

2. Sketching RD textures using the interface

Initially, the user sketches ellipses and lines in the sketch window, which act as inputs to the system. The system then generates a texture consisting of spots or stripes that has similar attributes as the initial sketch. These attributes are shape, size, spacing, and orientation of the spots and stripes that form the textures.

The sketch window is divided into smaller regions to allow for a pattern with varied attributes across the entire image. For the purpose of experimenting, the sketch was divided into four regions (Figure 2), but this could be *n by n* regions for a larger texture. Average values of attributes in each region are used in determining the final attributes of the output texture. Since patterns in different regions can vary in size, shape, or orientation, linear interpolation of the attribute values between neighboring regions gives the output texture a smooth transition from one region to the next.

2.1. Sketching spots

2.1.1. Controlling size of spots. Spots are described by their size, shape and orientation. The user can generate spots by sketching ellipses in the sketch window. The system calculates the average size of ellipses in each region. A mapping function then maps these values to a set of starting parameters such that the spots generated by this system will have the same size as those sketched. This mapping function is described later in Section 5.4.1. As shown in Figure 2, the spots are varied in size across the image with a smooth transition in size from the top left to the bottom right corner.

2.1.2. Controlling orientation of spots. Using the same technique as that for varying the size of spots as explained above, their orientation can also be varied across the image (Figure 3). Implementation details for generating textures with arbitrarily oriented spots can be found in Section 5.3.

2.1.3. Controlling spacing among spots. In general, a RD system of spots would have the spacing between spots determined by the size of spots. Implementing cascaded RD system as described in [Tur92], our system allows the user to generate patterns from a sketch where arbitrary sized spots can have arbitrary spacing between them (Figure 4). This technique will be described in detail in Section 4.

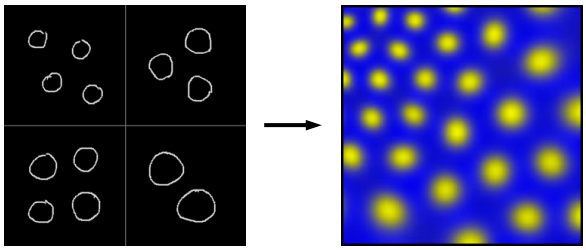


Figure 2. *Size of the spots produced varies over the grid (right), as depicted in user's sketch (left).*

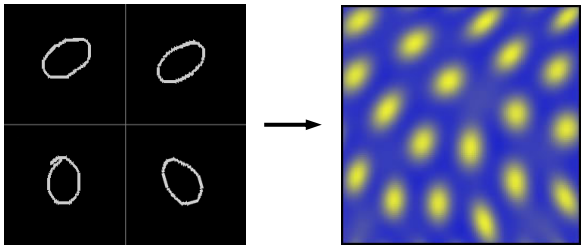


Figure 3. *Orientation of spots produced (right) varies over the grid as sketched (left).*

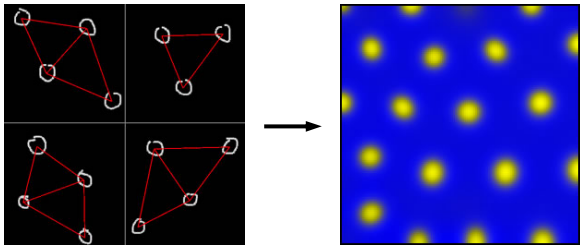


Figure 4. *Small spots with large spacing (right) as sketched in the sketch window (left). The red lines are produced when the sketch is analyzed. These lines correspond to the distances among the spots.*

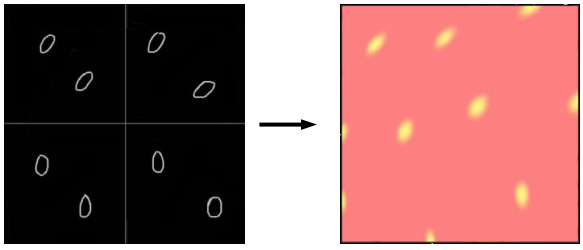


Figure 5a. *The first simulation produces yellow spots (right) based on the sketch of the larger ellipses (left).*

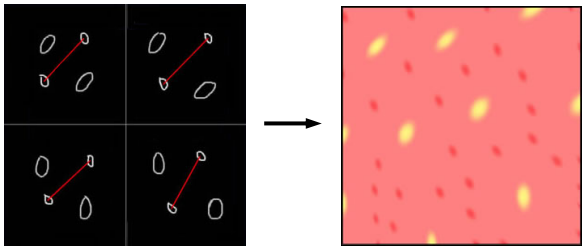


Figure 5b. *The second simulation produces red spots (right). Red lines between smaller ellipses in the sketch are produced when the sketch is analyzed (left).*

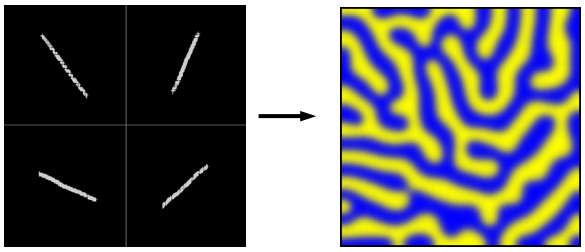


Figure 6. *The system generates pattern of stripes (right) that have the same orientation as in user's sketch (left).*

2.1.4. Complex patterns. The previous subsections described how spots with desired size, orientation and spacing can be obtained. However, the spots generated in this way have only one size in each region. In order to have small spots interspersed with larger spots through out the image (Figure 5b), we implement cascaded RD system as described in [Tur92].

Using our interface, the user first sketches only the large ellipses in the sketch window (Figure 5a). The system generates spots with spacing and orientation as sketched. These spots form the first layer of patterns, with a coloring scheme as described in Section 2.3. To add more spots into this texture, the user first freezes these larger spots (set the morphogens concentration in these cells constant). A second set of ellipses can now be drawn in the sketch window (Figure 5b). The system produces a second set of spots (the second layer of patterns), with orientation as sketched, in between the first set. The outcome of this second set of spots depends on the spacing among the first set and the size of the second set. The size of the second set of spots should be small enough for them to form in between the larger spots.

2.2. Sketching stripes

Stripe pattern can be generated by sketching straight lines in the sketch window (Figure 6). Stripes are described by their width and orientation. The user needs to draw at least one line representing stripes in each region of the sketch window. The system calculates average slopes of lines in each region. Based on these values, the system then generates stripes with orientations similar to those in the sketch. The detailed description of this implementation can be found in Section 5.1. The user can choose one of four different sizes to draw the line, resulting in four different thicknesses of the stripes.

2.3. Color control

Our system generates textures by imitating the RD process between morphogens. The concentration of each morphogen is mapped to a color. The patterns of spots and stripes are formed through the variations in concentration of the morphogens as a result of the RD process. After the texture is produced, the user can use the color control interface provided by the system to vary the color of the texture by changing the mapping of the morphogen concentrations to colors.

The color control interface is based on the work presented in [KTBG03]. The interface provides two sets of colors illustrated by the two long spacing bars in Figures 7 and 8. The upper bar displays the set of colors used for the first layer of patterns while the lower bar displays the set of colors used for the second layer (this second color set is used only in the case of complex patterns, as in Figure 8). A user can use the Red/Green/Blue slide bars to define the colors to be used for either of the two sets. Colors can be added, modified, removed or the order of colors can be changed for each of the two sets. A user can also adjust the spacing between colors in one set by moving the buttons (one for each color) on the spacing bar. Apart from the spacing bar, the user can use a Bezier curve to blend between colors. The blending of colors in each set is based on the distance between two adjacent colors on the spacing bar, and by the shape of the corresponding Bezier curve.

The concentration of a given morphogen is mapped to the continuous set of colors; the lowest concentration (valley) corresponds to the first color in the set, and the highest concentration (peak) corresponds to the last color (Figure 7).

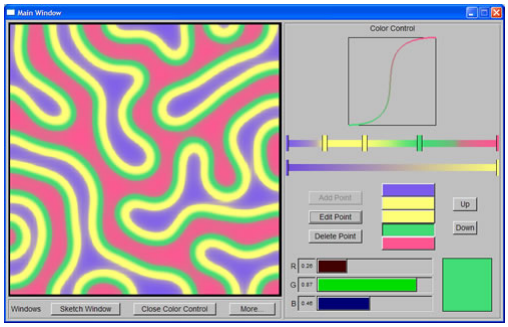


Figure 7. *Concentration of morphogen A is mapped to the upper set of colors, with the lowest concentration mapped to violet and the highest concentration mapped to pink. Transition between the valley to the peak of the morphogen's concentrations is visualized by a change in color, from violet to yellow, to green, and then to pink.*

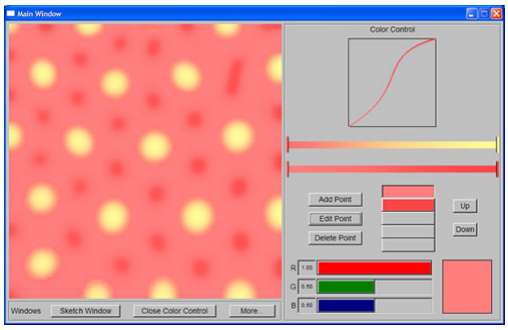


Figure 8. *Spots belonging to the first layer are mapped to the first color set (pink to yellow), while spots belonging to the second layer are mapped to the second color set (pink to red).*

3. Background of Reaction-Diffusion

A Reaction-Diffusion (RD) system consists of two (or more) morphogens *diffusing* over a surface and *reacting* with one another to form a stable pattern. This pattern is visualized by mapping each morphogen's concentration to a color. In general, this process can be modeled by the following differential equations [Tur91]:

$$\frac{\partial a}{\partial t} = F(a,b) + D_a \nabla^2 a$$
$$\frac{\partial b}{\partial t} = G(a,b) + D_b \nabla^2 b \qquad \text{(Eq. 1)}$$

where a and b represent the concentrations of morphogens A and B, and D_a and D_b are diffusion rates of A and B, respectively. At any point in time, the concentrations a and b are determined by the above functions. $F(a,b)$ and $G(a,b)$ describe the change in the concentration of A and of B, respectively, as a result of the reaction processes between A and B. These functions will decide if one morphogen inhibits the other, or sustains the other at the same position. Laplacians $\nabla^2 a$ and $\nabla^2 b$ are the measures of how high the concentrations of A and B are relative to the concentrations of the same morphogens, respectively, in the surrounding cells. In the case of a grid (Figure 9a), the formula for $\nabla^2 a$ is as follows:

$$\nabla^2 a = a_{i-1,j} + a_{i+1,j} + a_{i,j-1} + a_{i,j+1} - 4a_{i,j} \qquad \text{(Eq. 2)}$$

In general, for an arbitrary surface, e.g. a mesh (Figure 9b), the formula will be:

$$\nabla^2 a = \sum_{k=1}^{n} a_{ik} - n a_i \qquad \text{(Eq. 3)}$$

For a given morphogen A, if the concentration of A in the current cell is higher than that of A in the surrounding cells, Laplacian $\nabla^2 a$ is negative. In the next time step, A diffuses away from this cell, thus the concentration of A reduces. The reverse occurs if the concentration of A is lower in the current cell relative to the surrounding cells. D_a and D_b specify how fast A and B diffuse across the surface. At any time t, the concentration of A is the sum of a reaction process between A and B – denoted by $F(a,b)$ – and a diffusion process of A – denoted by $D_a \nabla^2 a$ (see Appendix).

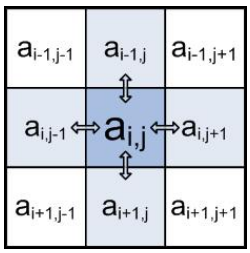


Figure 9a. *Morphogen A diffuses to and from four adjacent cells.*

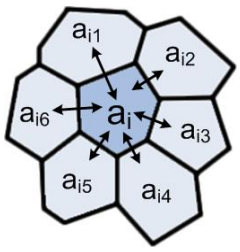


Figure 9b. *Morphogen A diffuses to and from all adjacent cells.*

4. Previous work

The work by Turing in [Tur52] describes a RD system of two morphogens that forms spots (Turing spot system). Other researchers have since shown that simple patterns, from spots to stripes, can also be generated with reaction-diffusion mechanisms using different systems of morphogens and equations [Bar81, Mur82, Mei82]. In [Mei82], Meinhardt described a RD system of two morphogens that produces spots (Meinhardt spot system), and another RD system of five morphogens that produces stripes (Meinhardt stripe system). In 1991, Witkin and Kass adapted RD for texture synthesis and added anisotropy by varying diffusion across the surface [WK91]. It was also shown that complex patterns could be generated using double simulations of cascaded RD systems [Tur91].

Using the three RD systems (Turing spot, Meinhardt spot, and Meinhardt stripe), we can generate random spots or stripes by applying random variations to the concentrations of morphogens in the system. We can also generate regular stripes by raising the initial concentration of one morphogen at certain cells, called "initiator cells." During RD simulation, stripes radiate from these cells (Figure 10).

Cascaded systems built upon these three basic forms can generate more complex patterns, such as leopard spots or mixed spots of different sizes [Tur92]. Mixed spots of different sizes are obtained by first generating large spots. These large spots are then frozen – the concentrations of morphogens in those cells that form the spots are kept constant. The second simulation then creates small spots in between the large spots (Figure 11a). Leopard spots, or rosettes (Figure 11b), are generated using the Turing spot system in a similar way, but with one extra step. After freezing the large spots, the concentrations of morphogens in cells that form these spots are reset to the initial level [Tur92]. During the second simulation, the small spots tend to form a circle around the large spots instead of in between them.

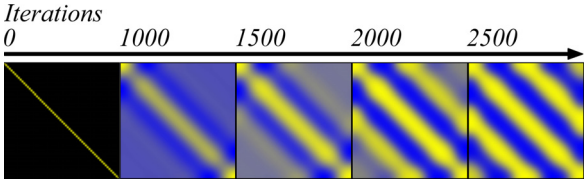


Figure 10. *Regular stripes radiate from initiator cells during RD process.*

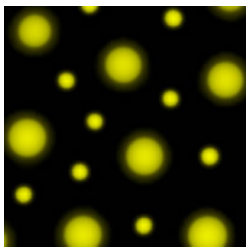


Figure 11a. *Mixed spots generated by the Meinhardt spot system.*

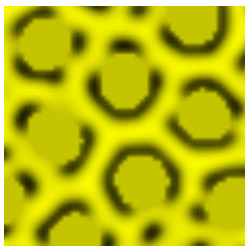


Figure 11b. *Leopard spots produced by Turing spot system.*

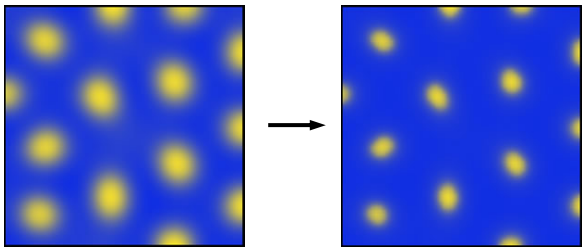


Figure 12. *Two-step generation of small spots with large spacing using cascaded Turing spot system. First, large spots with large spacing are produced (left). Next, small spots are produced at the same sites where large spots had been (right).*

When single RD simulation is employed to generate spots, the distance between spots, or spacing, is proportional to the size of the spots. However, using cascaded Turing spot systems as described in [Tur92], small spots with large spacing can be generated (Figure 12). The first simulation yields large spots with large spacing. The concentration of morphogens from the first simulation is then used to set up the second simulation such that smaller spots are produced at the same positions where the large spots had been, thus maintaining large spacing.

The existing systems of spots and stripes described above have limited flexibility. For example, the spots and stripes described before cannot be oriented in any arbitrary direction that the user may desire. More importantly, the user needs to directly tweak parameters of the RD system in order to get a desired pattern. It can be difficult to find initial values of morphogens that produce interesting patterns, much less a pattern that has certain desired attributes. In the following section we describe our interface with which a user can sketch RD texture. This is an intuitive method for a user to generate patterns automatically, without the need for knowledgeable tweaking of RD parameters. In order to allow more flexibility to the textures that can be generated, we also provide extensions that support arbitrarily oriented spots and stripes. In the following section, we will show how we give additional control to the attributes of the patterns, and how we apply machine learning to map from our high-level spot description to RD parameters.

5. Implementation

5.1. High-level description for spots

The main objective of this work is to develop an intuitive and easy way for a user to automatically generate a desired texture. The first step is the intuitive sketch interface described in Section 2. However, the RD systems which are used to generate the textures require values of various parameters as inputs. In order to extract the parameter values from the sketch, we devised a high-level description for the spots in terms of the three parameters: size, orientation and spacing.

When the user draws spots in the sketch window, the system fits ellipses to the spots, and extracts their centers, orientations and radii. The area of a spot is used to describe its size; the ratio of minor radius to major radius to describe its shape, and the major axis to describe its orientation. We then use Delaunay triangulation method to determine the closest neighbors of a spot, and then calculate the distance from this spot to its neighbors. The average distance between adjacent spots in the user's sketch is then used as the spacing between all the spots for the purpose of generating the texture.

5.2. Application of Machine learning for mapping user's sketch to RD parameters

In the previous section, we describe how physical attributes of the sketch are calculated. Once these values have been extracted, we use machine learning to map between the physical attributes and RD parameters. This step is described in the following paragraphs.

5.2.1. Preparing training data for machine learning. The purpose of our machine learning function is to map the spot attributes to parameter values. The first step is to learn how a set of initial parameter values map to spot attributes. Since the parameters used to determine orientation and size with spacing are independent, learning the mapping for the orientation and for the size with spacing can be done separately. To reduce the total number of data points, we generated four different sets of training data, two for Turing and Meinhardt spots' orientation and two for Turing and Meinhardt spots' size and spacing. The RD system was run 15,600 times with different parameter settings to generate textures with various patterns that serve as training data for this learning process. For each RD spot system, we use a different threshold value for morphogen A or B to trace the boundary of the spot. We then fit ellipses to these spots. Bad textures (e.g. when the patterns are not well formed, or if the size of the spots is too small to perform ellipse fitting) are excluded. Using the same method as for extracting spot attributes from user's sketch, we calculate the physical attributes of spots generated by the RD system (Figure 13). During the texture generation process, the training data were fed to a machine learning function to map the spot attributes from the user's sketch to corresponding RD parameters.

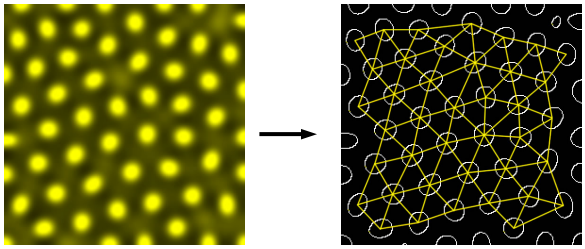


Figure 13. *Ellipses (left) are fitted to RD spots (right). Yellow lines connect each ellipse to their closest neighbors as determined by Delaunay triangulation.*

5.2.2. Locally Weighted Regression. To implement machine learning, we choose Locally Weighted Regression (LWR) because it does not require an extremely large collection of training data, and yet it is highly efficient for learning complex mappings from real-valued input vectors to real-valued output vectors using noisy data.

Locally Weighted Regression is a memory-based algorithm for learning continuous curves that uses only training data close to the particular point X. Points nearby are weighted by their distance to point X. A nonlinear regression – an estimation technique using interpolation to predict one variable from one or more other variables – is then calculated using these weighted points. We use the LWR Matlab function presented in [SA94].

5.2.3. Mapping functions. The results we get from mapping size and spacing of spots to RD parameters are not always reliable because of the random nature of the Reaction-Diffusion technique. Therefore, we build a mapping function for size and spacing of spots using linear interpolation. The function takes spacing or area as input and returns reaction rate as output (one to one relationship). For orientation of spots, the mapping function is a LWR function that takes as inputs the ratio of minor radius to major radius as well as the two (x & y) components of the major axis vector of the sketched ellipses, and returns direction coefficients as outputs.

Inputs	Outputs	Mapping functions
Spacing	Reaction rate (k or s)	Linear Interpolation
Area	Reaction rate (k or s)	Linear Interpolation
Ratio of minor radius to major radius, x component of major axis vector, y component of major axis vector	Direction coefficients (E-W, S-N, SE-NW, SW-NE)	Locally Weighted Regression

Table 1. *Input sketch parameters with their related output RD parameters and mapping functions.*

5.3. Extensions to existing RD spots

5.3.1. Arbitrarily oriented spots. The existing RD systems described in the Section 4 can generate only anisotropic spots, which are elliptical spots in vertical or horizontal directions. This is because these systems allow the morphogens to diffuse to four neighbors only, along the directions North, South, East, and West (Figure 9a). In this work, apart from using anisotropic diffusion rates, we also allow the morphogens to diffuse from and to each of the eight neighbors (Figure 14a). In addition to diffusion rates D_a and D_b shown in Equation 1, we also expand the formula calculating Laplacians (Equation 2) and add direction coefficients to the flow of morphogens to and from neighbor cells (Figure 14a). These coefficients specify preferential directions in which morphogens diffuse faster, thus the spots are elongated in these directions. As a result, spots can have arbitrary orientation.

Even though morphogens can flow from and to all eight directions, they always flow equally in symmetrically opposite directions. Therefore, four direction coefficients (E-W, S-N, SE-NW and SW-NE) are enough to define the diffusion of chemicals to all eight neighboring cells.

5.3.2. Non-homogeneous patterns. In a RD system, there are several parameters such as the reaction rates, diffusion rates, and direction coefficients that determine the size, shape, spacing, and orientation of the spots. In order to allow non-homogeneous patterns, instead of using the same parameter values everywhere, we blend the parameter values across the grid. The result is patterns of spots that have size, orientation, and spacing changing across the image (Figure 15).

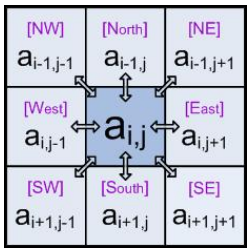


Figure 14a. *Morphogens from one cell can diffuse from and to all eight of its neighbors. Direction coefficients give the spots shape and orientation.*

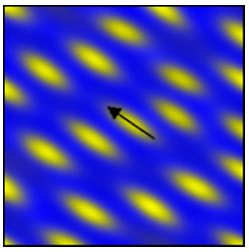
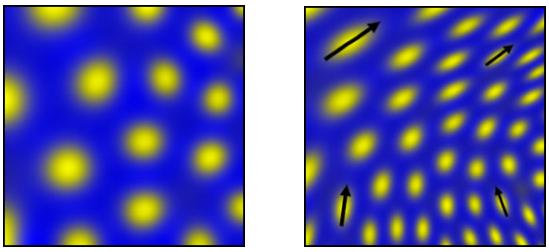


Figure 14b. *Anisotropic Turing spots with the user-supplied direction indicated by arrow*



(a) **(b)**

Figure 15. *Examples of non-homogeneous textures of spots with size (a) and orientation (b) changing across the image.*

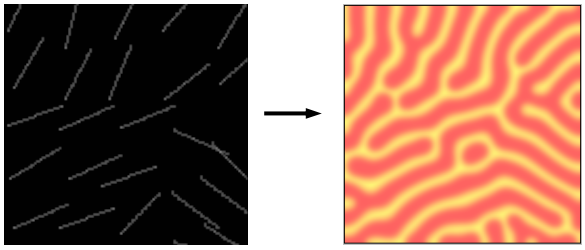


Figure 16. *Seeding the stripe system with line segments (left) results in a texture with similar orientation (right). Note that slopes of the stripes vary across the image.*

5.4. Oriented random stripes

Direction coefficients can be used to generate oriented spots (as explained in Section 5.3.1), but they do not help in generating oriented stripes. Initiator cells that form oriented random line segments, however, can create oriented random stripes. The system calculates the average slopes of the lines sketched by the user in each of the regions of the sketch window. These values are used to seed random lines with slight variations of these average slopes. The Meinhardt stripes system then generates random stripes that have similar orientations (Figure 16).

6. Sketching RD texture on a 3D mesh

We demonstrate our interface for creating RD textures on 3D surfaces. Instead of drawing on a separate sketching window, the user can draw spots directly on a mesh. The system then fits ellipses to the spots, computes their attributes, maps these to appropriate parameters, which are then associated with vertices of the mesh that lie within these ellipses. The remaining vertices are assigned parameter values that are a linear combination of the values at vertices lying within the closest ellipses.

To generate spots of different sizes on the bunny in Figure 19, the diffusion rates of morphogens A and B are propagated and blended over the mesh based on the size and position of spots sketched by the user. The relative proportions of morphogen flowing to each of the neighboring vertices are calculated as explained below.

For each vertex i in the mesh and its neighboring vertex ik (among its n neighboring vertices numbered $i1$ through in), the term d_{ik} denotes the distance between centroids of the two triangles on either side of the edge connecting vertex i to vertex ik (Figure 17). The percentage coefficient P_{ik} for the flow of morphogens between these two vertices is calculated as the ratio:

$$P_{ik} = \frac{d_{ik}}{\sum_{k=1}^{n} d_{ik}} \qquad \text{(Eq. 4)}$$

And the Laplacian $\nabla^2 a$ for vertex i is given by Equation 5, where a_i is the morphogen concentration in vertex i.

$$\nabla^2 a = 4 \times \left(\sum_{k=1}^{n} P_{ik} a_{ik} - a_i \right) \qquad \text{(Eq. 5)}$$

To generate oriented spots, the vector showing the orientation of the ellipse is projected onto the tangent plane at each vertex (Figure 18). During RD simulation, the faces surrounding each vertex are rotated so that this projected vector becomes parallel to the x axis. These faces are then scaled down along the x axis according to the ratio of the minor radius to the major radius of the ellipse sketched by the user. Finally, the relative proportions of morphogen flowing to each of the neighboring vertices are determined based on these transformed faces surrounding the vertex in the same manner as described above. Once the differential flow of morphogens has been calculated, it is applied to the original mesh to generate the desired pattern. As shown in Figure 20, the method produces spots with desired orientation on a cube. However, due to irregular distribution of vertices, the orientation of spots is not as clear when generated on the bunny.

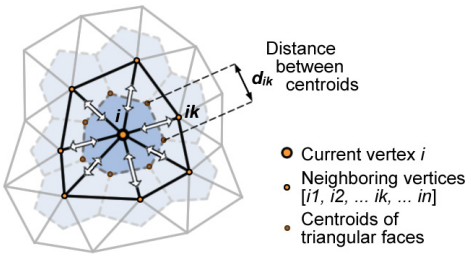


Figure 17. *Proportional flow of morphogens along each edge between the current vertex and its neighboring vertices is determined by the normalized distances between the centroids of the two faces on either side of that edge.*

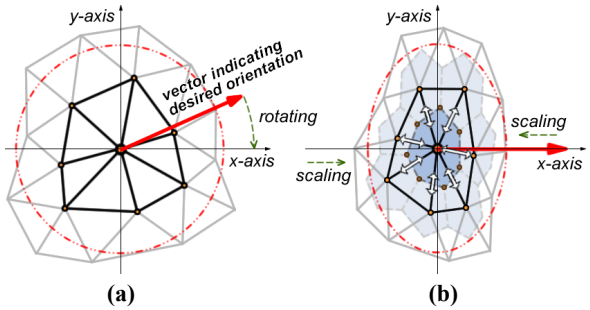


Figure 18. *(a) The original mesh showing the current vertex and its surrounding vertices. The arrow shows the projection of the sketched ellipse's major axis vector onto the tangent plane at the current vertex. (b) Illustration of the vertices after a rotation followed by scaling down the faces along the x-axis.*

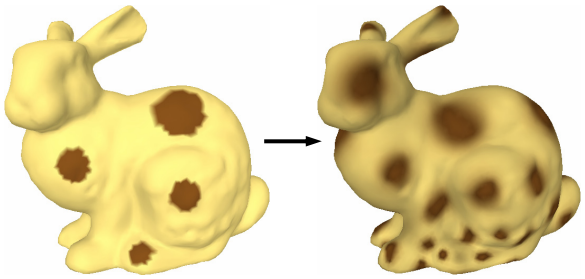


Figure 19. *Spots of different sizes generated on a 3D mesh.*

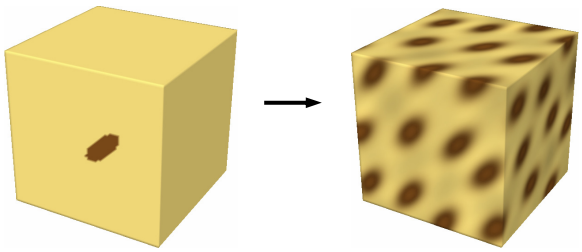


Figure 20. *Oriented spots generated on a cube.*

7. Summary and future work

The system described here allows a user to make a simple sketch of a desired pattern and then automatically generates a large amount of texture that has a similar pattern. This interface allows the user to produce a desired texture with ease and without the need for any internal knowledge of the Reaction-Diffusion process. Instead of having to run the system several times to find the right set of parameters that will produce desired texture, the user can sketch the patterns they want and have the system automatically generate a texture with similar attributes. This is a much more natural and intuitive method.

As part of future work, the dimension and resolution of the grid can be dynamically determined based on user's sketch, such as number and size of ellipses or stripes, in order to make the texture synthesis more flexible. As demonstrated here, this system can be extended to support automatic texture synthesis on canonical 3D surfaces. The next step will be the development of a stable and robust system for automatic texture synthesis on a 3D model.

8. Appendix:

8.1. Formulae for Turing spot system

$$\frac{\partial a}{\partial t} = speed\left[k_a(16 - ab) + D_a\nabla^2 a\right]$$
$$\frac{\partial b}{\partial t} = speed\left[k_a(ab - b - \beta) + D_b\nabla^2 b\right]$$

8.2. Formulae for Meinhardt spot system

$$\frac{\partial a}{\partial t} = speed\left[s\left(\frac{0.01a^2\beta}{b} - ap_1 + p_3\right) + D_a\nabla^2 a\right]$$
$$\frac{\partial b}{\partial t} = speed\left[s\left(0.01a^2\beta - bp_2 + p_3\right) + D_b\nabla^2 b\right]$$

8.3. Formulae for Meinhardt stripe system

$$\frac{\partial a}{\partial t} = speed\left[\frac{0.01a^2e\beta}{c} - ak_{ab} + D_a\nabla^2 a\right]$$
$$\frac{\partial b}{\partial t} = speed\left[\frac{0.01b^2d}{c} - bk_{ab} + D_b\nabla^2 b\right]$$
$$\frac{\partial c}{\partial t} = speed\left[0.01a^2e\beta + 0.01b^2d - ck_c\right]$$
$$\frac{\partial d}{\partial t} = speed\left[(a - d)k_{de} + D_d\nabla^2 d\right]$$
$$\frac{\partial e}{\partial t} = speed\left[(b - e)k_{de} + D_e\nabla^2 e\right]$$

9. References:

[Bar81] BARD J. B. L.: A Model for Generating Aspects of Zebra and Other Mammalian Coat Patterns. *Journal of Theoretical Biology*, Vol. 93, No. 2, pp. 363–385 (November 1981).

[EMP*02] EBERT D. S., MUSGRAVE F. K., PEACHEY D., PERLIN K., WORLEY S.: *Texturing & Modeling: A Procedural Approach*, Morgan Kaufmann, 3rd edition, (December 2002).

[KTBG03] KULLA C. D., TUCEK D. J.., BAILEY R., GRIMM C. M.: Using Texture Synthesis for Non-Photorealistic Shading from Paint Samples. *11th Pacific Graphics Conference on Computer Graphics and Applications,* pp. 477 (October 2003).

[Mal82] MALDELBROT B. B.: *The Fractal Geometry of Nature*, W. H. Freeman and Company, New York, 1982.

[Mei82] MEINHARDT H.: *Models of Biological Pattern Formation*, Academic Press, London, 1982.

[Mur81] MURRAY J. D.: On Pattern Formation Mechanisms for Lepidopteran Wing Patterns and Mammalian Coat Markings. *Philosophical Transactions of the Royal Society B*, Vol. 295, pp. 473–496 (October 1981).

[Per85] PERLIN K.: An Image Synthesizer. *Computer Graphics*, Vol. 19, No. 3 (SIGGRAPH '85), pp. 287–296 (July 1985).

[PH89] PERLIN K., HOFFERT E. M.: "Hypertexture," *Computer Graphics*, Vol. 23, No. 3 (SIGGRAPH '89), pp. 253–262 (July 1989).

[SA94] SCHAAL S., ATKESON C. G.: Assessing the quality of learned local models. *Advances in Neural Information Processing Systems 6* Morgan Kaufmann, San Mateo, CA, pp. 160-167, 1994.

[Tur52] TURING A.: The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society B*, Vol. 237, pp. 37–72 (August 14, 1952).

[Tur91] TURK G.: Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion. *Computer Graphics*, Vol. 25, No. 4 (SIGGRAPH '91), pp. 289–298 (July 1991).

[Tur92] TURK G.: Texturing Surfaces Using Reaction-Diffusion. PhD Dissertation, The University of North Carolina at Chapel Hill, 1992.

[WK91] WITKIN A., KASS M.: Reaction-Diffusion Textures. *Computer Graphics*, Vol. 25, No. 4 (SIGGRAPH '91), pp. 299–308 (July 1991).