Sketching Reaction-Diffusion Texture

Ly Phan
Southern Arkansas University

Cindy Grimm
Washington University in St. Louis

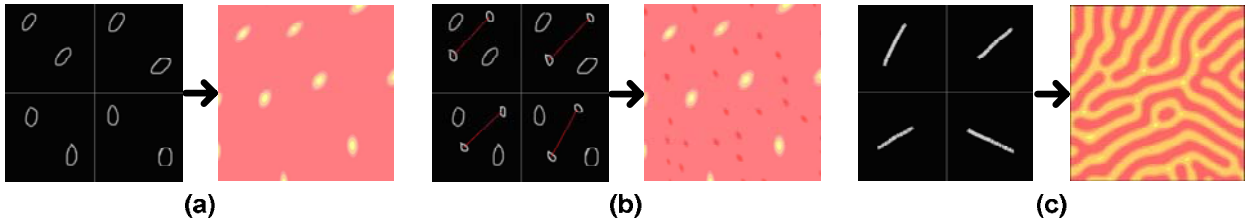

Figure 1. *(a) User sketches ellipses with different orientation. System generates dots with similar size, shape, and orientation. (b) User adds smaller ellipses to the sketch. System generates smaller dots in between larger dots. (c) User sketches lines with different slopes. System generates random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

Abstract

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 a great flexibility for mapping textures to arbitrary surfaces. However, it can be difficult to find starting values of parameters that will produce good patterns. We use machine learning to resolve the major difficulty of determining appropriate initial values of the RD system. The system described here allows a user to sketch a pattern, 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 other patterns as well as manipulate the color of the resulting texture.

I. Introduction

Textures are two-dimensional or three-dimensional images that can be mapped onto an object's surface in many ways, including color mapping, bump mapping, etc. 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 – or bitmapped – 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 a particular type of procedural textures called Reaction-Diffusion textures. This approach has the advantage of allowing an unlimited amount of non-repeating texture and offers a great flexibility for controlling the attributes of patterns and for 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 chemicals form certain animal coat patterns, such as spots and stripes [Turing 52]. This process is defined by two or more differential equations, each of which controls the change over time in concentration of one morphogen with respect to that of other morphogens.

A RD system consists of numerous parameters, such as the initial concentrations of chemicals, the reaction rates and diffusion rates, etc., that will be described in greater details in section III. At present there is no intuitive way to determine the values of these parameters that will produce good patterns, much less a pattern that has certain desired attributes. Hence, although RD can produce interesting and varied textures, it is difficult for a user to use an RD system without having a deep knowledge of the system. We solve this problem by developing an interactive interface that allows users to sketch a rough idea of the pattern they want. The system then automatically generates a texture sample that has the same features (as shown in 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. This method allows for more control over the attributes of the patterns produced by the RD system, for example the size and orientation of spots and stripes. The program interface also includes a color system that gives the user much more freedom to vary the coloring of the generated textures.

The rest of this paper is organized as follows. In Section II, we present the interface for sketching RD textures, illustrating what the user sees and does. In Section III, we provide a brief background of RD systems. In Section IV, we summarize the previous work that has been done on generating textures with RD. Section V contains a detailed description of the implementation of this system, including our extension to the existing techniques and our application of machine learning to solve the problem of finding the right parameter values.

II. Sketching RD textures using the interface

There are two steps to generate a RD texture from a sketch. In the first step, the user sketches ellipses and lines in the sketch window, which act as inputs to the system. In the second step, the program automatically generates an output that is a texture of spots or stripes that has the same attributes as the sketch. These attributes are shape, size, spacing, and orientation of the spots and stripes that form the textures.

The sketch window provided by the program is divided into smaller regions to allow for a pattern with varied attributes across the image. For the purpose of experimenting, it was divided into four regions (as shown in Figure 2), but it could be *n by n* regions for a larger texture. Only average values of attributes of patterns sketched within one region are used in deciding the attributes of the output texture. As 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.

II. 1. Sketching spots

a. Controlling size of spots

The user can generate spots by sketching ellipses in the sketch window. Spots are described by their size, shape and orientation. After the user sketches ellipses, the program calculates the average area 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 V.4.c. 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.

b. Controlling orientation of spots

Using the same technique, the spots produced can also have varied orientation across the image as shown in Figure 3.

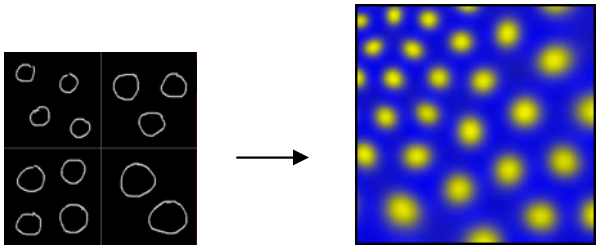

Figure 2. *Area of the spots produced varies over the grid, as depicted in user's sketch.*

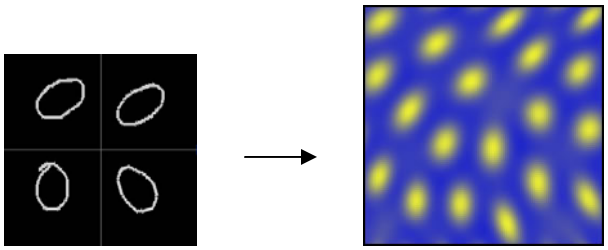

Figure 3. *Orientation of spots varies over the grid*

c. Cascaded systems

Similarly, the program can generate a texture of spots with specified size and spacing as shown in Figure 4. The implementation of this procedure is described in detail in Section IV.

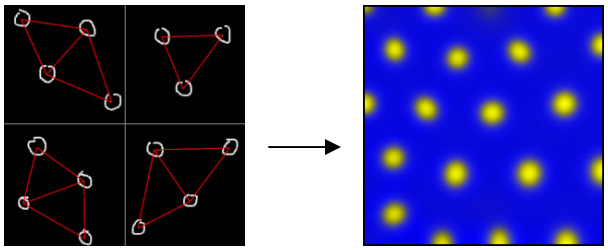

Figure 4. *Small spots with large spacing as sketched. Red lines are produced when the sketch is analyzed.*

d. Layered patterns

The system can also generate spots with different sizes that have both the spacing and the orientation as sketched. To do this, the user first needs to draw larger ellipses in the sketch window. The system generates these spots with the spacing and orientation as specified in the user's sketch, as shown in Figure 5a. To add more spots into this texture, the user first freezes these larger spots. A second set of ellipses can now be drawn in the sketch window. The system produces a second set of spots that also have the orientation specified in the second sketch, as shown in Figure 5b. This second set of spots are formed, or 'layered,' in between the first set of spots. The outcome of this second set of spots depends on the spacing among the first spots and the size of the second spots. The size of the second spots should be

small enough for these spots to form in between the larger spots.

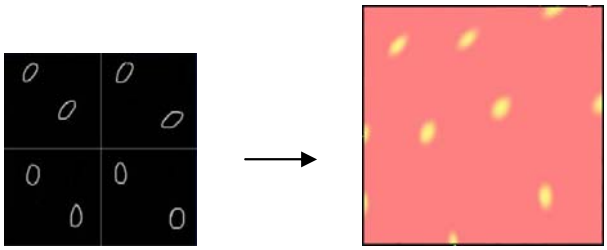

Figure 5a. *The first simulation produces yellow spots based on the sketch of the larger ellipses.*

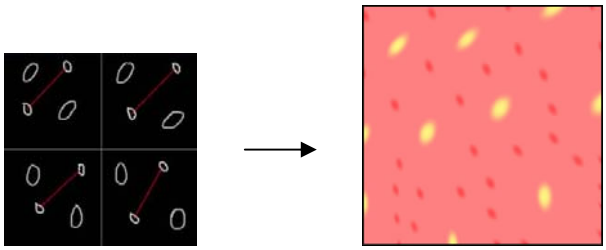

Figure 5b. *The second simulation produces red spots. Red lines between smaller ellipses are produced when the sketch is analyzed.*

II. 2. Sketching stripes

Stripe pattern can be generated by sketching straight lines with a particular orientation in the sketch window as shown in 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 the orientations varied across the image as in the sketch. The detailed description of this implementation can be found in section V-1. The user can choose one of four different sizes to draw the line, resulting in four different thicknesses of the stripes.

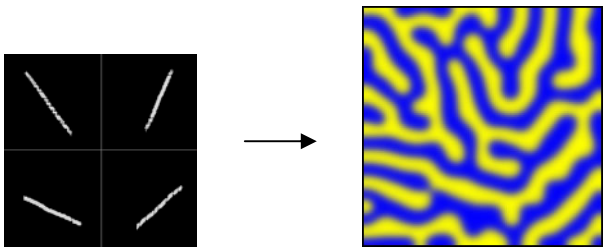

Figure 6. *Pattern of stripes generated based on user's sketch.*

II. 3. Color control

The program generates textures by imitating the RD process between chemicals, also called morphogens. The concentration of each morphogen is mapped to a color. The patterns of spots and stripes are formed by the variations in chemical concentrations as a result of the RD process. After the texture is produced, the user can use the color control provided by the program to vary the color combination of the texture by changing how the chemical concentrations are mapped to colors.

The color control interface based on the work presented in [Kulla and Bailey 03] provides two sets of colors. The upper set of colors is used for the first layer of patterns, the lower one for the second layer of patterns (this set is used only in textures with layered patterns). A user can define a color and then add it to either set or modify the original color. The user can also remove a color from a color set, or change the order of colors in a set. The color sets are visualized by two spacing bars, on which there are buttons – one for each color – that the user can move back and forth to change the spacing among colors. Apart from the spacing bar, an implementation of Bezier curve also allows the user to shape the way one color transforms into another. The transition of colors in each set is decided by the distance between two adjacent colors on the spacing bar, and by the shape of the corresponding Bezier curve.

The chemical concentration is mapped to the continuous set of colors; the lowest concentration (0) corresponds to the first color in the set, and the highest concentration (1) corresponds to the last color. In Figure 7 below, concentration of chemical *a* is mapped to the upper set of colors, with the lowest concentration mapped to violet and the highest concentration mapped to pink. A transition from the valley to the peak of the chemical concentration is visualized by a transition in color, from violet to yellow, to green, and then to pink.

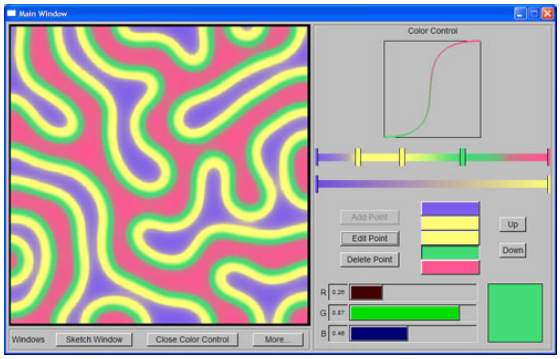

Figure 7. *Color control interface maps the concentration of a chemical to a set of colors, allowing user greater freedom to manipulate the texture.*

For layered patterns, the chemical concentrations can be mapped to two different sets of colors; the upper set

is mapped to the first layer of patterns, and the lower mapped to the second layer of patterns.

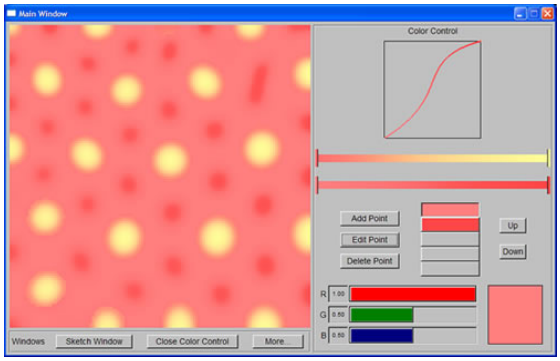

Figure 8. *Spots belonging to the first layer are mapped to the first color set (pink to yellow), while spots belonging to the second layer are mapped to the second color set (pink to red).*

III. Background of Reaction-Diffusion

The method for generating textures presented in this paper is based on Reaction-Diffusion process. A Reaction-Diffusion (RD) system can consist of two (or more) chemicals *diffusing* over a surface and *reacting* with one another to form a stable pattern. This pattern is visualized by mapping each chemical's concentration to a color. In general, this process can be modeled by the following differential equations [Turk 91]:

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

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*. It is these functions that decide whether one chemical will inhibit the other, or will sustain 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 in comparison with the concentrations of the same chemicals, respectively, in the surrounding cells. In the case of a grid, the formula for $\nabla^2 a$ is as follows:

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

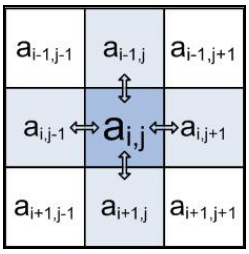

Figure 9. *Morphogen A diffuses to and from four adjacent cells.*

In general, for an arbitrary surface, e.g. a mesh, the formula will be:

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

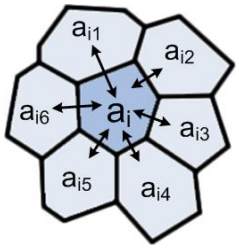

Figure 10. *Morphogen A diffuses to and from all adjacent cells.*

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, thus the concentration of *A* in this cell decreases in the next step of time, which means *A* diffuses away from this cell. On the contrary, if the concentration of *A* is lower than that of the surrounding cells, $\nabla^2 a$ is positive, thus the concentration of *A* in this cell increases in the next step of time, that is, *A* diffuses toward this cell. The same explanation applies for $\nabla^2 b$. D_a and D_b are the diffusion rates for *A* and *B*, respectively, that 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 $D_a \nabla^2 a$. Similarly, at any time *t*, the concentration of *B* is the sum of a reaction process between *A* and *B* – denoted by *G(a,b)* – and a diffusion process of *B* – denoted by $D_b \nabla^2 b$.

IV. Previous work

The work by Turing in [Turing 52] describes a RD system of two chemicals 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 chemicals and equations [Bard 81, Murray 82, Meinhardt 82]. In 1991, Witkin and Kass adapted RD as a method for texture synthesis and

added anisotropy by varying diffusion across the surface [Witkin and Kass 91]. In the same year, it was shown that complex patterns could also be generated using double simulations of cascaded RD systems [Turk 91].

Three basic reaction-diffusion systems that are described in [Turk 92] were originally proposed were originally proposed by Turing [Turing 52] and Meinhardt [Meinhardt 82]. These systems are referred to in this paper as Turing spot, Meinhardt spot, and Meinhardt stripe, using the name of the authors and the forms the systems produce, respectively.

As the names of these RD systems indicate, we can generate random spots or stripes (as shown in Figure 7) by giving random variations to the concentrations of chemicals in the system. Another method to generate regular stripes is to raise the initial concentration of one chemical at certain special cells, called "initiator cells." During RD simulation, stripes radiate from these cells, as demonstrated in Figure 11 below.

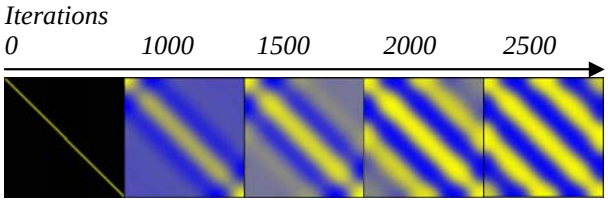

Figure 11. *Regular stripes radiates from initiator cells during RD process.*

When we build cascaded systems from these three basic forms, we can also generate more complex patterns (layered patterns), such as leopard spots or mixed spots of different sizes [Turk 92]. The generation of some of these complex patterns is now described. In the following section we will describe more extensions of our own that provide the user with more flexibility in the textures that can be generated.

Mixed spots of different sizes are obtained by first generating large size spots. These large spots are then frozen -- the concentrations of chemicals in those cells that form the spots are kept constant. The second simulation then creates small spots in between the large spots as shown in Figure 12.

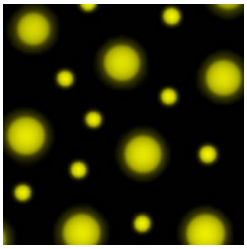

Figure 12. *Mixed spots generated by the Meinhardt spot system*

Leopard spots, or rosettes, as shown in Figure 13, are generated using the Turing spot system in a similar way, but with one extra step. After freezing the large spots, the concentrations of chemicals in cells that form those spots are reset to the initial level [Turk 92]. During the second simulation, since the concentration of morphogen *A* in the large spots is much lower than its peak, *A* tries to diffuse to cells around the large spots and, at the same time, inhibits the concentration of morphogen *B* from rising in these cells. As a result, the small spots tend to form a circle around the large spots instead of in between them.

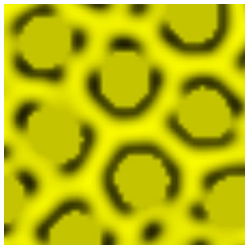

Figure 13. *Leopard spots by Turing spot system*

When single RD simulation is employed to generate spots, the distance between spots, or spacing, is proportional to the size of the spots. The larger the spots are, the wider the spacing is. However, small spots with large spacing, as shown in Figure 13, can be generated by using cascaded Turing spot systems. The first simulation yields large spots with large spacing. The concentration of chemicals 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 have been, thus maintaining large spacing. In other words, the first simulation produces spots with the specified spacing. The second simulation yields spots with the desired size while maintaining the previous spacing.

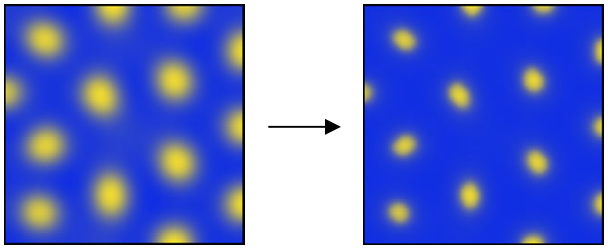

Figure 14. *Turing spots with large spacing*

V. Implementation

The existing systems of spots and stripes described in the previous section 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 chemicals that produce good patterns, much less a pattern that has certain desired attributes. In this section we describe how we implement a sketching interface with which a user can sketch RD texture. This is an intuitive and easy way for a user to generate patterns automatically. In order to provide more flexibility to the textures that can be generated, we also provide some extensions of our own that support arbitrarily oriented spots and stripes. We also show how we add more control to the attributes of the patterns, and how we apply machine learning to map from our high-level spot description to RD parameters.

V. 1. Implementation for oriented random stripes

Direction coefficients can be used to generate oriented spots, but they do not help in generating oriented stripes. Initiator cells that form oriented random line segments, however, can create oriented random stripes. To do this, the system first analyzes the slopes of the lines drawn by the user. It then calculates the average slopes for each of the regions of the sketch window. These values are used to seed random lines with slight variations of predetermined slopes over the texture. The Meinhardt stripes system then generates random stripes that have similar orientations as in the sketch as shown in Figure 15.

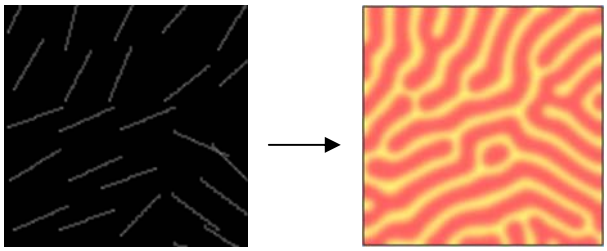

Figure 15. *Seeding the stripe system with line segments (left) results in a texture with similar orientation (right). Slopes of the stripes vary across the image.*

V. 2. Extensions to existing RD spots

a. Implementation for arbitrarily oriented spots

The existing RD systems described in previous section can generate only anisotropic spots, which are elliptical spots in vertical or horizontal directions. This is because these systems allow the chemicals diffuse to four neighbors only, along the directions North, South, East, and West, shown in Figure 9. In this work, we allow the chemicals to diffuse to each of the eight neighbors, as shown in Figure 15a, together with anisotropic diffusion rates, thus we can make spots with any orientation. Chemicals in each cell can diffuse from and to all eight neighbors of the cell. In addition to diffusion rates D_a and D_b shown in Equation 1, we also add direction coefficients to the diffusion part of the RD functions, specifying preferential directions with faster diffusion rates. Under the effect of these coefficients, the chemicals diffuse faster in certain directions than in others, thus the spots are elongated in these directions. As a result, spots can have arbitrary orientation.

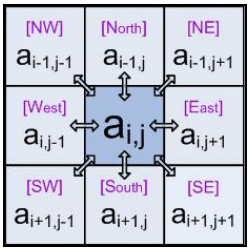

Figure 16a. *Morphogens from one cell can diffuse from and to all eight of its neighbors. Diffusion coefficients give the spots shape and orientation.*

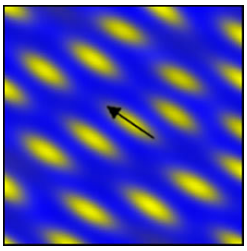

Figure 16b. *Anisotropic Turing spots with the user-supplied direction indicated by arrow*

b. Implementation for 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 allow these parameters to have different values across the image. The result is patterns of spots that are varied in size, orientation, and spacing, as shown in Figure 17 below.

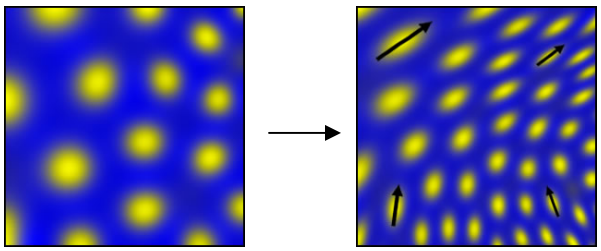

Figure 17. *Examples of non-homogeneous textures composed of spots with changing size and orientation*

V. 3. 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 in this direction is the easy-to-use sketch window already described in Section II. However, the RD systems which are used to generate the textures need the values

of various parameters such as diffusion rates, reaction rates, etc. as inputs. In order to extract the parameter values from the sketch that the user has drawn, we devised a high-level description for the textures in terms of the three parameters size, orientation and spacing.

For each RD spot system, we use a threshold value for chemical A or B to trace the boundary of the spot. The locations of the cells on the edge of the spot are passed into an ellipse-fitting tool. We fit ellipses to the spots, and extract 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 use Delaunay triangulation method to determine the closest neighbors of a spot, and then calculate the distance from this spot to the neighboring centroids. 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.

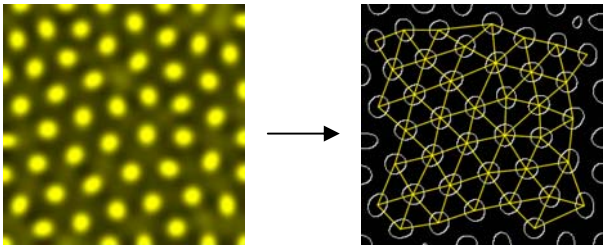

Figure 18. *The fitted ellipses (circles) and their closest neighbors are shown as determined by Delaunay triangulation.*

V. 4. Application of Machine learning for mapping user's sketch to RD parameters

In the previous section, we describe how physical attributes of the pattern are calculated. Once these values have been extracted from the user's sketch, they need to be mapped to the parameters, such as reaction rates, diffusion rates, etc. In order to resolve the mapping between physical attributes and RD parameters, we use the well-known method of machine learning. This step is described in the following paragraphs.

a. Preparing training data for machine learning

The physical attributes calculated in the previous step are used as training data for a machine learning function. The data generation task also includes detecting bad textures (e.g. when the patterns are not well formed, or when the area of the spots is smaller than 7 cells). Since orientation of spots and size or spacing of spots are determined by different sets of parameters, learning the mapping for the orientation and for the size and 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 program was run 15,600 times with different parameter settings to generate training data for machine learning function. The training data points were then fed to a machine learning function to map the sketch attributes to correspondent RD parameters.

b. 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 point X in question, as shown in Figure 19. 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. Using a memory-based algorithm for machine learning also means that we need to store the training data and run LWR function every time to generate the parameters. The LWR Matlab function that we used for this task was presented in [Schaal and Atkeson 94].

c. Mapping functions

The results we get from mapping size and spacing of spots to RD parameters are not always reliable because of the erratic nature of data. 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 rates and direction coefficients as outputs. For orientation of spots, the mapping function is a LWR function that takes the ratio between minor and major radii and the major vector of the sketched ellipses as inputs, and returns direction coefficients as outputs.

VI. 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 the same pattern. This interface allows any user to produce a desired texture with unprecedented ease and without the need for any deep knowledge of the Reaction-Diffusion process. Instead of having to run the system numerous times to find the right set of

parameters that will produce texture with the desired attributes, the user can sketch the patterns he/she wants and have the system automatically generate a texture with similar attributes, which is a much more natural and intuitive method.

In order to make the texture synthesis more flexible, 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. This system can also be extended to support automatic texture synthesis on arbitrary mesh. The next step will be automatic texture synthesis on 3D model.

VII. Appendix:

A. Formula for Turing spot system:

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

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

B. Formula for Meinhardt spot system:

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

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

C. Formula for Meinhardt stripe system:

$$\frac{\partial a}{\partial t} = speed\left[\frac{0.01a^2e\beta}{c} - ak_{ab} + D_a\nabla^2 a\right]$$

$$\frac{\partial b}{\partial t} = speed\left[\frac{0.01b^2d}{c} - bk_{ab} + D_b\nabla^2 b\right]$$

$$\frac{\partial c}{\partial t} = speed\left[0.01a^2e\beta + 0.01b^2d - ck_c\right]$$

$$\frac{\partial d}{\partial t} = speed\left[(a - d)k_{de} + D_d\nabla^2 d\right]$$

$$\frac{\partial e}{\partial t} = speed\left[(b - e)k_{de} + D_e\nabla^2 e\right]$$

VIII. References:

[Bard 81] Bard, Jonathan 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).

[Ebert et al 02] Ebert, David S., F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley, *Texturing & Modeling: A Procedural Approach*, Morgan Kaufmann, 3rd edition, December 2002.

[Kulla and Bailey 03] Kulla, Christopher D., James D. Tucek, Reynold Bailey, Cindy Grimm, "Using Texture Synthesis for Non-Photorealistic Shading from Paint Samples," *11th Pacific Graphics Conference on Computer Graphics and Applications* (2003)

[Maldelbrot 82] Maldelbrot, Benoit B. (1982), *The Fractal Geometry of Nature*, W. H. Freeman and Company, New York, 1982.

[Meinhardt 82] Meinhardt, Hans, *Models of Biological Pattern Formation*, Academic Press, London, 1982.

[Murray 81] 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.

[Perlin 85] Perlin, Ken, "An Image Synthesizer," *Computer Graphics*, Vol. 19, No. 3 (SIGGRAPH '85), pp. 287–296 (July 1985).

[Perlin and Hoffert 89] Perlin, Ken and Eric M. Hoffert, "Hypertexture," *Computer Graphics*, Vol. 23, No. 3 (SIGGRAPH '89), pp. 253–262 (July 1989).

[Schaal and Atkeson 94] Schaal, Stefan and Christopher G. Atkeson, "Assessing the quality of learned local models," *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, San Mateo, CA, pp. 160-167 (1994).

[Turing 52] Turing, Alan, "The Chemical Basis of Morphogenesis," *Philosophical Transactions of the Royal Society B*, Vol. 237, pp. 37–72 (August 14, 1952).

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

[Turk 92] Turk, Greg, "Texturing Surfaces Using Reaction-Diffusion," *PhD Dissertation*, The University of North Carolina at Chapel Hill, 1992.

[Witkin and Kass 91] Witkin, Andrew and Michael Kass, "Reaction-Diffusion Textures," *Computer Graphics*, Vol. 25, No. 4 (SIGGRAPH '91), pp. 299–308 (July 1991).