Texture Synthesis for Data Visualization Using Spot Noise

Chen-Chun Lai
lai@cs.cornell.edu
Computer Science Department
Cornell University
Ithaca, New York 14850

Advisor: Dr. Bruce Land

August 2, 1994

Abstract

Using spot noise to synthesize textures is a useful technique for data visualization. Based on Jarke J. van Wijk's paper, an IBM Visualization Data Explorer module *SpotNoiseTexture* was implemented in this project. Spot noise texture synthesis technique allows easy local control, and a wide variety of textures can be synthesized. Local control of the texture could be realized by variation of the spot. The *SpotNoiseTexture* module provides local controls such as spot orientations, sizes and styles, noise types, mean values and standard deviations. Various examples of spot noise texture are shown in this paper.

Contents

List of Figures

1 Introduction

Current texture synthesis techniques for data visualization do not provide local control for various scalar and vector fields over surfaces. Most of them are not suitable for the design of texture. A new technique, *spot noise*, was presented by Jarke J. van Wijk [7] for efficient synthesis with local control and ease of texture design.

Spot noise is synthesized by addition of randomly weighted and positioned spots. Local control of the texture can be realized by variation of the spot. Unlike other texture design techniques, spot is simply a visual primitive directly related to the texture being synthesized. This feature provides a straight forward sense to design a texture. We'll examine four simple styles of spots with various local controls to synthesize textures in this paper.

If we want to visualize a 2-D flow velocity field with a texture, such as the dominant direction that aligns with the direction of the flow, it's simple to generate this visual image by applying some line-style spots along flow directions to a random noise field using spot noise synthesis technique. We'll see some sample textures using this technique for an electric field.

2 Spot Noise Texture Synthesis for Data Visualization

2.1 Spot Noise

Spot noise is the spatial analogue of shot noise. It is produced by successive repetition of independent pulses at random intervals. If each pulse is generated by the profile $a_i h(t - t_i)$, thus the resultant noise function $f(t)$ can be expressed as

$$f(t) = \sum_i a_i h(t - t_i),$$

where the the indepent time values t_i form a random sequence.

The power spectrum of $f(t)$ is directly related to the energy spectrum

$$S_h(\omega) = |H(\omega)|^2$$

of $h(t)$, where $H(\omega)$ is the Fourier transform of $h(t)$.

If a_i has zero mean, and on average there are v repetitions per unit time, then the power spectrum $P_f(\omega)$ of $f(t)$ is

$$P_f(\omega) = v < a_i^2 > S_h(\omega),$$

where $< a_i^2 >$ represents the mean of a_i^2.

The spatial analogue also has many applications. For the application discussed here, the pulse $h(x)$ is considered as a spot that is dropped on the plane, and we call this noise spot noise. The size of a spot is limited, and usually small compared to the size of the texture segment to be synthesized. In analogy with shot noise, the spot noise is defined as

$$f(x) = \sum_i a_i h_i(x - x_i),$$

where x_i are random positions on the plane. If on average there are v repetitions per unit area then

$$P_f(k) = v < a_i^2 > S_h(k),$$

where k is the two-dimensional frequency vector.

2.2 Spot Noise Texture Synthesis

The power spectrum of the texture and the energy spectrum of the spot are the same, except for a scale factor. So, realizations of spot noise can be constructed in the frequency domain via the multiplication of the fourier transform $H(k)$ with a scale factor and with a random phase shift α_k to $H(k)$. The effect of a random phase shift α_k is equivalent to multiplication with

$$W(k) = e^{i\alpha_k}.$$

The power spectrum of $w(x)$ is evenly distributed over all frequencies, so $w(x)$ is white noise. According to the convolution theorem, multiplication in the frequency domain is equvalent to convolution in the spatial domain, hence spot noise can also be synthesized via convolution of $h(x)$ with white noise.

Variation of the texture for data visualization can be realized via variation of the spot. This requires a variable spot $h(p, x)$, whose properties are controlled by a set of parameters p. These parameters are determined via a data mapping m from the data $d(x)$ that belong to the texture coordinates x. Spot noise for data visualization can thus be synthesized by using variable spots:

$$f(x) = \sum_i a_i h(m(d(x_i)), x - x_i).$$

In this project, the *SpotNoiseTexture* DX module applies a special convolution of variable 2D spot function $Spot(p, x, y)$ with a noise field to synthesize spot noise textures. The spot noise function is extended to be

$$f(x, y) = \sum_i \sum_j Noise_{i,j} Spot(m(d(x, y)), x - x_i, y - y_i).$$

3 Implementation of *SpotNoiseTexture* DX Module

In this project, spot noise texture synthesis is implemented to a DX module under IBM Visualization Data Explorer Version 2.0. The *SpotNoiseTexture* module is composed of the following four parts:

- Input processor
- Noise generator
- Spot generator
- Convolution processor

3.1 Input processor

There are six input parameters for the *SpotNoiseTexture* module. The first two inputs are automatically processed with type recognition. Either scalar field or vector field data can be interpreted correctly as corresponding orientation controls and size controls.

- **Input 1: Spot Orientation Controls – M x N field**
  - scalar x: Orientation, in radian
  - 2D vector $[x, y]$: Orientation, as $\tan^{-1}(\frac{y}{x})$
- **Input 2: Spot Size Controls – M x N field**
  - scalar x:
    - for line : $Length = x$
    - for ellipse : $Radius = x$ (a circle)
    - for rectangle: $Width = Height = x$
    - for cross : $Width = Height = x$
  - 2D vector $[x, y]$:
    - for line : $Length = x$
    - for ellipse : $R1 = x, R2 = y$
    - for rectangle: $Width = x, Height = y$
    - for cross : $Width = x, Height = y$
- **Input 3: Noise μ (mean value) – float value**

  Default value is zero mean. (μ=0.)
- **Input 4: Noise σ (standard deviation) – float value**

  Default value is 0.2. (σ=0.2.)
- **Input 5: Spot Type – string**

  Default spot type is ”line”.

  - ”line” (default) : controlled by length and orientation
  - ”ellipse” : controlled by long axis, short axis and orientation
  - ”rectangle” : controlled by width, height and orientation
  - ”cross” : controlled by width, height and orientation

- **Input 6: Noise Type – string**

  Default noise type is ”gaussian”.

  - ”white” : noise in unform distribution. (white noise)
  - ”gaussian” : noise in Gaussian distribution.

3.2 Noise generator

There are two noise types available in the *SpotNoiseTexture* module, white noise and Gaussian noise.

White noise is evenly distributed over all frequencies.

Gaussian noise is generated by Polar method for normal deviates. [4] The algorithm is:
1. Get uniform variables v$_1$, v$_2$ which are uniformly distributed between -1 and +1.
2. Let $s = v_1^2 + v_2^2$.
3. If $s \geq 1$ then go back to step 1.
4. Compute x, where $x = v_1 - 2\sqrt{\frac{\ln s}{s}}$
5. return $noisevalue = \sigma \cdot x + \mu$

3.3 Spot generator

There are four basic styles of spot – line, cross, ellipse and rectangle.

All of the four spot generation algorithms provide free-angle rotation and adjustable size control. The scaling of a spot is limited within the maximum size 31x31. Except line spots, all the spots can be controlled by both width and height. Oversized portion of a spot is not generated outside the boundaries. These algorithms make solid fills for both ellipse and rectangle spots. See source code *spotnoise.c* for details.

- Lines are generated by a modified fast Bresenham line algorithm.
- Ellipse spots are generated by a simple general ellipse algorithm and inside filled.
- Cross are generated by two perpendicular lines.
- Rectangle spots are generated by four side lines, and inside filled.

3.4 Convolution processor

Spot noise texture synthesis is performed by a special convolution with position-oriented variable kernel filters (spots). The variations of spot are determined by the first two inputs (orientation and size controls) of the SpotNoiseTexture module. This special convolution algorithm implements the spot noise synthesis as

$$f(x,y) = \sum_i \sum_j Spot_{x,y}(i,j) Noise(x+i, y+j).$$

where $Spot_{x,y}$ is a spot generated by orientation and size controls on position (x, y), and $Noise$ is generated by μ and σ based on either white noise or Gaussian noise.

In the source code of the *SpotNoiseTexture* module (*spotnoise.c*), either linear convolution or circular convolution is possible. Default setting is for circular convolution.

3.5 The Output

The output of the *SpotNoiseTexture* module is a field which has synthesized spot noise texture in its "data" component. We can simply attach an *AutoGrayScale* module to produce a result image.

4 Examples of Spot Noise Textures

Some sample spot noise textures were synthesized by *SpotNoiseTexture* DX module.

- Figure 1 shows the results of applying different size controls on the four basic spot styles.
- Figure 2 shows the results of applying different orientation controls.
- Figure 3 shows visualized images synthesized by a simple sine wave data set with four basic spots.

5 Demostration

5.1 Files Description

In project directory *spotnoise*, there are full sources, documents and demostration files for this project.

- spotnoise.mak – Makefile for this project.
- spotnoise.mdf – Module description file for *SpotNoiseTexture* module.
- spotnoise.c – Source code for *SpotNoiseTexture* module.
- spotnoise.net, spotnoise.cfg – Demostration net file and its configuration file.
- efield.net, efield.cfg – A synthesized texture for an electric field data set.
- tiff/*.tiff – Sample spot noise texture images.
- document.ps – This document.
- READ.ME – A description to this project.

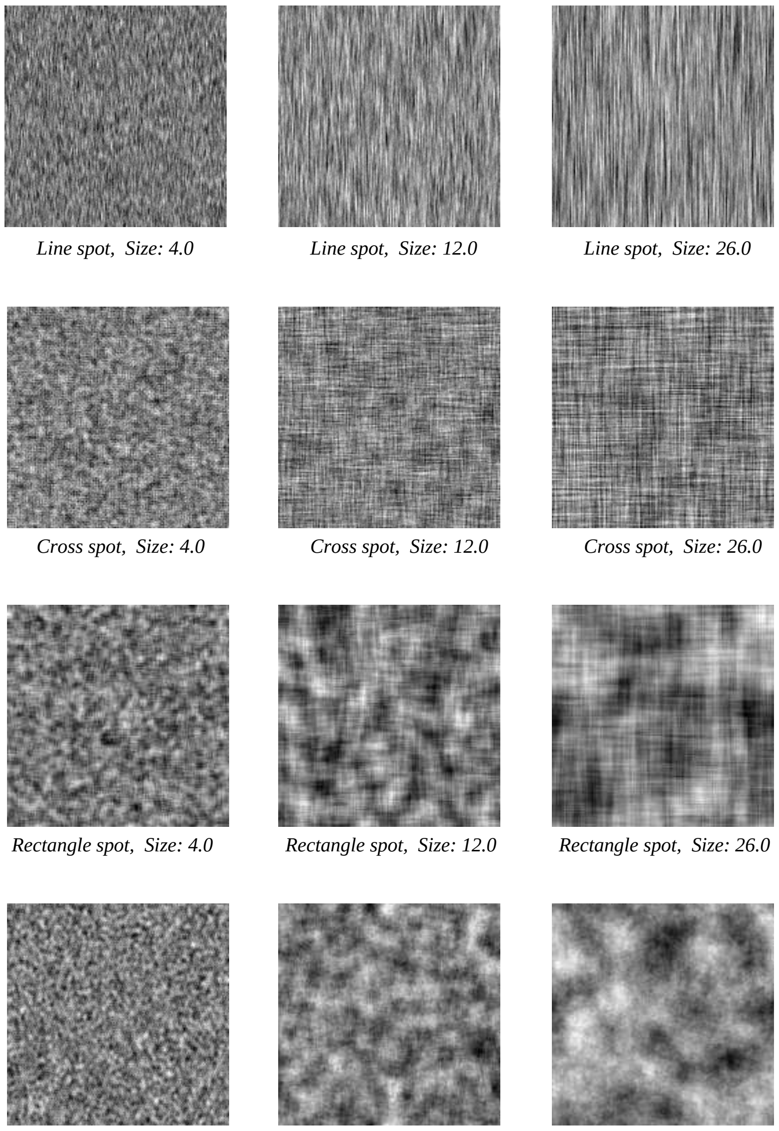

Figure 1: Applying various size controls with basic spots

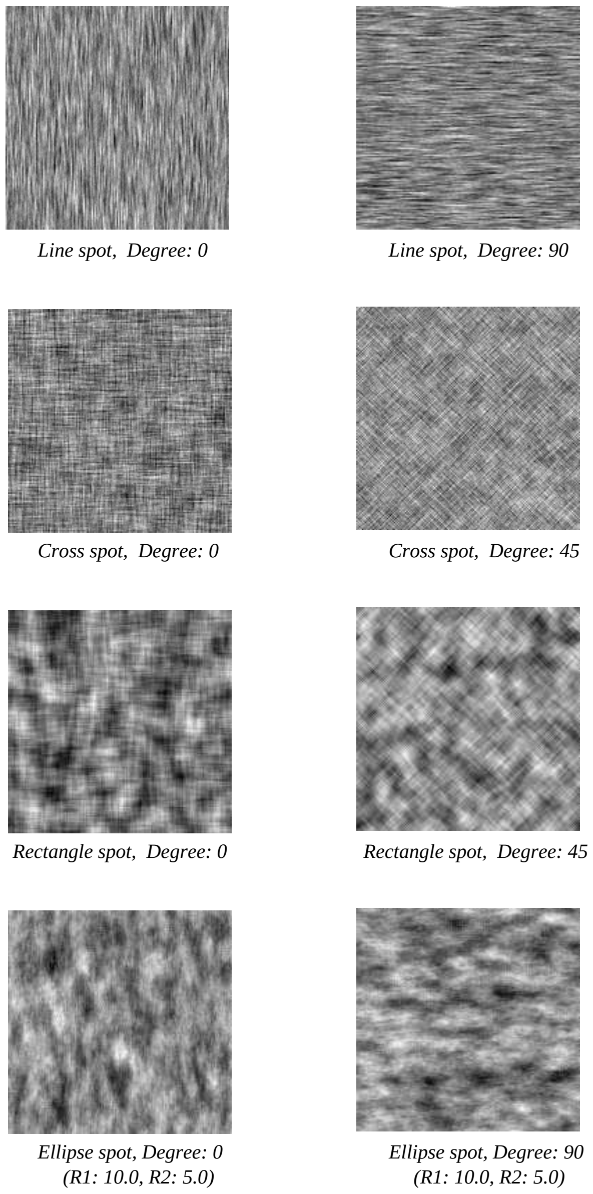

Figure 2: Applying various orientation controls with basic spots

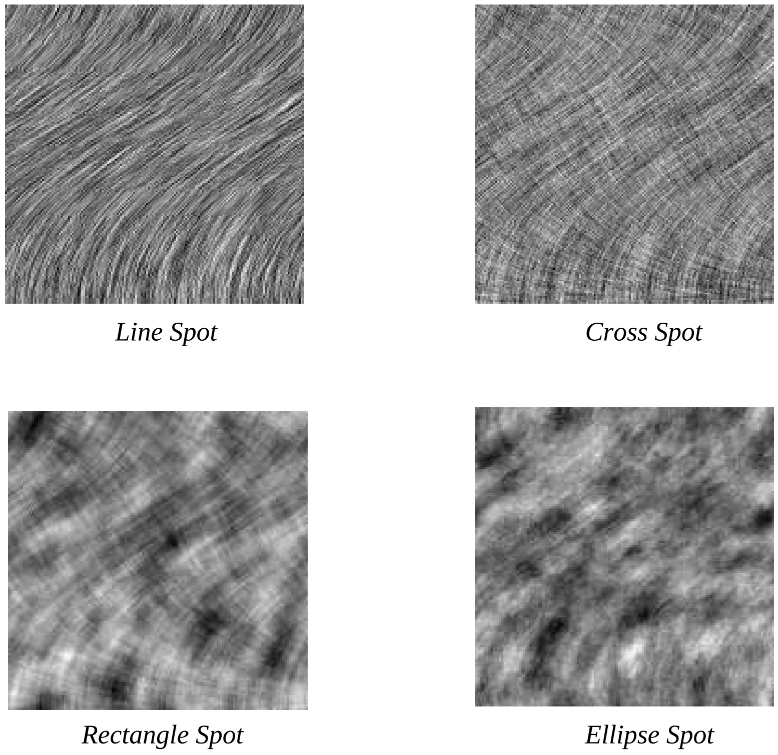

Figure 3: Simple sine wave data set visualized by basic spot noise textures

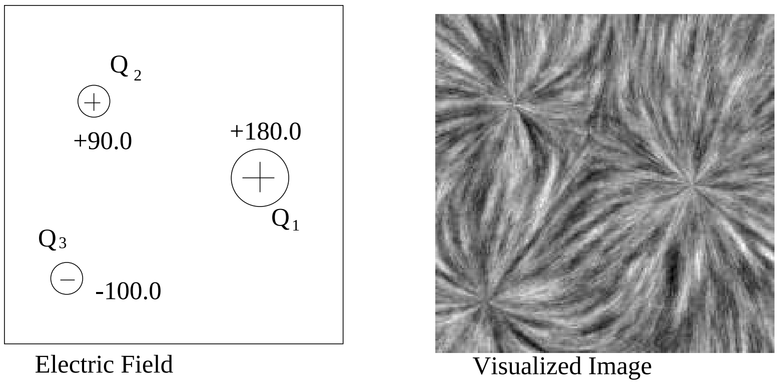

Figure 4: Synthesized image for an electric field data set

5.2 Running Demo Net: *spotnoise.net*

Before running the demo net, we have to link the new *SpotNoiseTexture* module into dxexec. Type

make -f spotnoise.mak

then bring up Data Explorer

dx -mdf spotnoise.mdf -exec ./dxexec

From *FILE* menu, *LOAD* demo net: *spotnoise.net*. Open CONTROL PANEL to modify some input parameters. Then select *EXECUTE* to synthesize a texture.

6 Simple Application – Visualize An Electric Field Intensity

A simple application using spot noise texture technique is to visualize an electric field intensity data set.

We built a electric field contains two positive electric point charge Q1, Q2 and one negative electric point charge Q3.

$$Q_1 : position = [95.0, 145.0], charge = +180.0$$
$$Q_2 : position = [140.0, 45.0], charge = +90.0$$
$$Q_3 : position = [30.0, 30.0], charge = -100.0$$

The electric field intensity E is defined as

$$E = \gamma \frac{Q}{4\pi\epsilon_0 r^2}$$

The visualized result is showed in Figure 4. Figure 5 shows various synthesized images of this electric field using different spot noise textures.

7 Conclusion and Further Works

The *SpotNoiseTexture* DX module is useful for scientific data visualization applications. For any 2D grid data set, we can simply apply local controls from input data to generate a visual image. By the local controls of spot orientations and sizes, noise types, noise mean and standard deviation values, we can easily synthesize various images for a data set with different textures.

The performance of the *SpotNoiseTexture* module could be enhanced by some optimization techniques. The spot generator could be optimized by faster raster algorithms. For example, a better general ellipse algorithm can enhance the performance if we choose ellipse spot to synthesize textures.

The process of spot noise texture synthesis is made via a special circular convolution with various spot filters. This process could also be made via addition of a random phase shift to the Fourier transform of the spot, followed by an inverse Fourier transform and normalization. This may enhance the synthesis performance if we use fast Fourier transform.

In this project, only four basic styles of spots are built in the *SpotNoiseTexture* module. It's possible to import user defined spot as spot input, then scale and rotate it by the size and orientation controls. This will make a more general, flexible texture generator and it may be easier to design a texture using user defined spots.

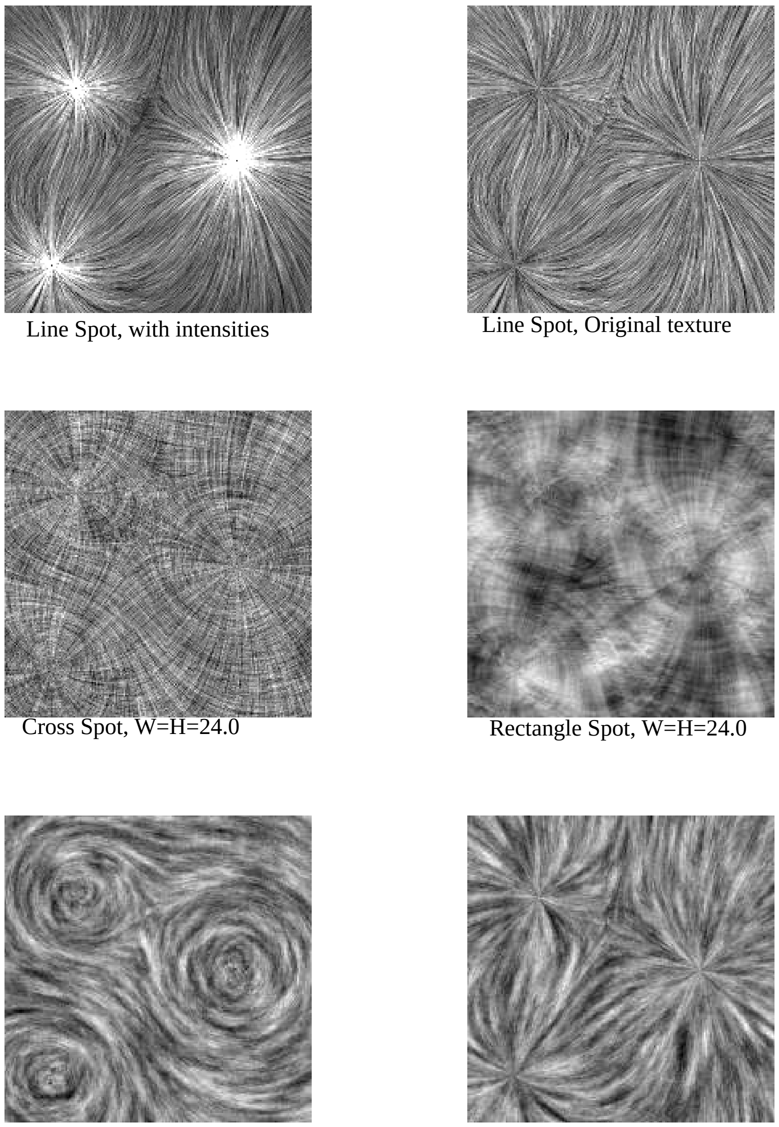
Line Spot, with intensities
Line Spot, Original texture
Cross Spot, W=H=24.0
Rectangle Spot, W=H=24.0
Ellipse Spot, R1=4.0 R2=24.0
Ellipse Spot, R1=24.0 R1=4.0

Figure 5: Various synthesized electric fields using basic spot noise textures

8 Acknowledgements

I would like to thank my advisor Professor Bruce Land for his valuable advice and constant encouragement throughout this project.

References

[1] Foley, J., Dam, A. V., Feiner, S., Hughes, J.,; *Computer Graphics Principles And Practice.* 2nd Ed., Addison-Wesley, 1990.

[2] IBM; *IBM Visualization Data Explorer User's Guide.* 4th Ed., Oct. 1993.

[3] IBM; *IBM Visualization Data Explorer Programmer's Reference.* 4th Ed., Oct. 1993.

[4] Knuth, D.E.; *The Art of Computer Programming.* Addison-Wesley, 1973.

[5] Kraus, J.D.; *Electromagnetics.* 3rd Ed., McGraw-Hill, 1984.

[6] Land, B.; Lecture notes, CS 417, CS 418. Computer Science Department, Cornell University, Spring 1994.

[7] Wijk, J.J. van. *Spot Noise, Texture Synthesis for Data Visualization*, Computer Graphics Volume 25, Number 4, July 1991, Page 309-318.