EG UK Theory and Practice of Computer Graphics (2005)
M. McDerby, L. Lever (Editors)

Perlin Noise and 2D Second-Order Tensor Field Visualization

Abstract

There has been much research in the use of texture for visulization the vector field data, whereas there has only been a few papers concerned specifically with tensor field data. This set is more complex and embeds more information than vector fields. In this paper, firstly texture is modeled by Perlin Noise. We show that by controlling the parameters of Perlin Noise, the user can control the output texture effectively, which is similar to Spot Noise. Then the modeled texture is used to visualize eigenvector fields of tensor fields by simple convolution. Several examples are shown. Compared to Line Integration Convolution, this method does not need to integrate the streamline along the vector field.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Picture/Image generation]: I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture

1. Introduction

Texture methods have recently been explored in many literatures, especially in the fluid flow field, LIC(Line Integration Convolution) [CL93] and Spot Noise [Van91] are the two most widely used texture methods. A good survey can be found in [LHD*04]. In the use of Perlin Noise [PH89], most of the past discussion has focused on how to produce realistic images rather than its application to data visualization. We have found Perlin Noise is an efficient way to visualize tensor fields. Furthermore, much research effort in texture visualization has been devoted to the vector field, whereas there are only a few research papers in the field of texture visualization of tensor fields. However, second-order tensor fields play a central role in many areas of physics and mechanical field, such as velocity gradient, stress and strain.

In this paper, we first discuss the modeling of Perlin Noise from the perspecitive of data visualization. Special emphasis is put on the control of the parameters of the Perlin Noise generation, which will have significant effect on the data visualization. Then we show that Perlin Noise has similar properties to the Spot Noise in the local control. After the modeling has been described, the tensor visualization is simply discussed and we applied Perlin Noise to the tensor field visualization in section 3 for a real application. Finally, in section 5 conclusions are drawn.

2. Perlin Noise

Perlin [PH89] [Per85] makes use of a single controllable stochastic *noise* function together with a toolkit of shaping functions to generate convincing representation of clouds, fire, water, stars etc.

Perlin defines the function $Noise(\mathbf{x})$ to generate solid textures through the composition of non-linear functions for stochastic textures, which is implemented as a summation of pseudo-random spline knots, one for each point on the integer lattice of $\mathbf{R}^2$. The knot at a regular grid (i,j) consists of a pseudo-random linear gradient $G_{i,j}$ weighted in each dimension by a smooth drop off function $\omega(t)$(see equation 1 and Figure 1) [PH89].

$$\omega(t) = \begin{cases} 2|t|^3 - 3|t|^2 + 1 & |t| < 1 \\ 0 & |t| \geq 0 \end{cases} \tag{1}$$

It can be regarded as a random number generator, while if you pass the same parameter twice, the same random number is produced.

We will discuss the resulting texture image from a designer's point of view. There are two issues of special interest—the size and direction properties. If scaling the modeling primitive $Noise(\mathbf{x})$ to $Noise(\text{scale}\mathbf{x})$, the user can control the output texture effectively. In Figure 2, from left to right, the first image's scale parameter is 10 times larger

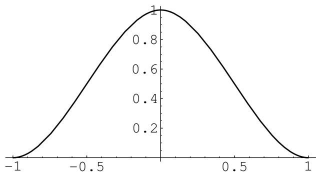

Figure 1: *The drop off function* $\omega(t)$

than the second one and 100 times larger than the third one, the output texture changes from a white noise like texture to a fractal like texture. This can be explained from a signal processing point of view. The scale parameter in fact corresponds to the frequency of the 2D signal. The bigger the scale is, the higher the frequency is. High frequency shows the detail within an image, and low frequency represents the bluring of an image. In Figure 3, if the scale parameter in the y direction is 10 times bigger than the x direction, i.e., the frequency of a 1D signal in the y direction is 10 times higher than the frequency of the signal in the x direction, the texture is stretched in the x direction.

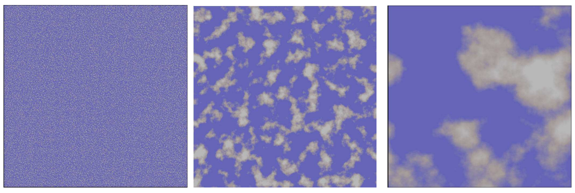

Figure 2: *Different scale of Perlin Noise*

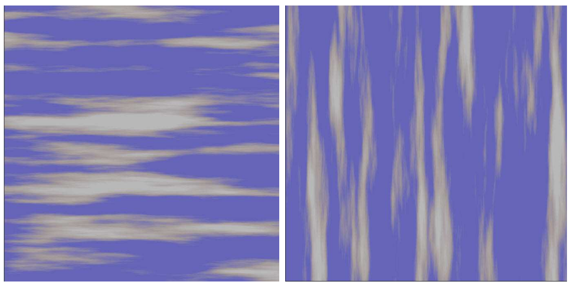

Figure 3: *Texture Pattern in different direction*

2.1. Comparison of Perlin Noise and Spot Noise

The issue of local variation of texture is discussed by van Wijk [Van91]. We found the above properties of Perlin Noise are shared with the Spot Noise. Spot Noise is defined as

$$f(\mathbf{x}) = \sum_i a_i h(\mathbf{x} - \mathbf{x}_i) \qquad (2)$$

where $\mathbf{x}_i$ are random positions on the plane, a_i is a random scaling factor with a zero mean. The pulse $h(\mathbf{x})$ is considered as a spot that is dropped on the plane. The size of the spot is limited, and usually small compared to the size of the texture segment to be synthesized. This method can be compared to the filtering of a very noisy image with the spot as the filter kernel. One of properties of the Spot Noise is that designer can control the generated texture by scaling the size of the spots(see Figures 4 and 5). In the Spot Noise, if small spots are used, samples at different locations are uncorrelated, and hence the result is white noise. Large spots degenerate to random faults, so the result will be fractal. These images are similar to those generated by Perlin Noise(see Figure 2 and Figure 3). But the bigger of size of spots is, the lower frequency of the resulting image is, which is contrary to the scale parameter in the Perlin Noise.

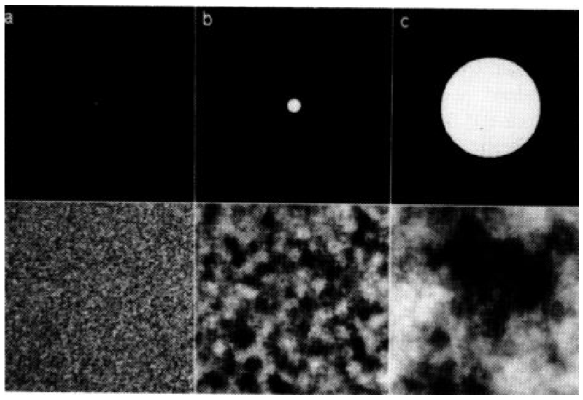

Figure 4: *Different sizes of spot [Van91]*

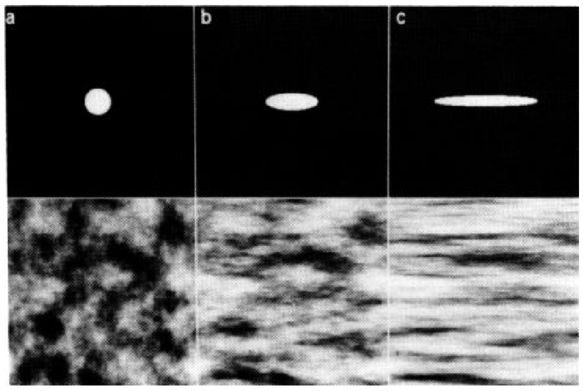

Figure 5: *Non-proportional scaling [Van91]*

3. Rail Residual Stress Tensor Data Visualization Using Perlin Noise

The tensor data, represented by equation 3, that we are interested in is namely residual stress in a Rail Head which is computed from measured strain (see Figure 6, model and data courtesy of J. Kelleher, Manchester Materials Science Center).

$$\begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} \tag{3}$$

Traditionally, this stress tensor can be represented by visual-

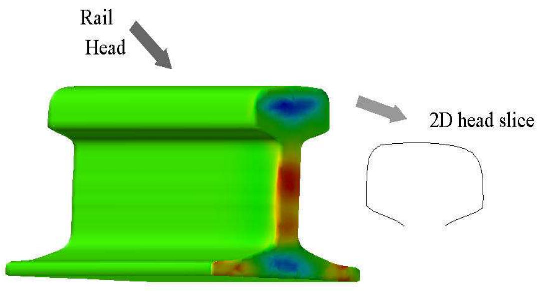

Figure 6: *The Rail Model, a slice of rail is extracted and the head of it is of interest*

izing the three different components of the stress. Therefore three images are necessary. In the 3D case, nine images are needed. Furthermore, every component of tensor data critically depends on the coordinates. With the transformation of coordinates, the tensor components are varied. However, the stress tensor is a physical quantity, so it should keep invariant in different coordinates.

Therefore, instead of visualizing stress tensor components directly, a stress tensor field can be decomposed to two eigen-vector fields: $\hat{\mathbf{v}}^{(1)}$ and $\hat{\mathbf{v}}^{(2)}$,i.e. two principal stresses vector fields. Principal stress fields are invariant in different coordinate and are equal to the original tensor field. There are already some glyphs to visualize two principal stress fields, such as ellipse or Haber glyph [Hab90]. One problem of those glyph visualizations is that they exist in a discrete manner, and can easily result in visual clutter for a large field. Therefore, the visualization techniques should be designed to reflect this continuity property and avoid the visual clutter.

The approach we use here was first discussed by Delmarcelle [Del94], who proposed Hyperstreamline icons to visualize 3D symmetric tensor data. In the 2D case, a hyperstreamline is a *stress trajectory* which is a line whose tangent at every point is in the direction of a principal stress. Then the state of the residual stress that varies through the rail head can be represented by a network of those stress trajectories.

4. Implementation

We develop Perlin Noise and its application in the rail tensor field as several modules in AVS/EXPRESS [UTFK*89], which provides easy manipulation of parameters of Perlin Noise to generate diffrent images. In this rail head application, the original tensor field is 19×20, which are linear interpolated to 1024×1024 so that the generated image size will be 1024×1024. Then a Jacobi method [PTVF02] is used to compute the eigenvector and eigenvalue of this tensor field. Therefore at every node of a 1024×1024 grid, there is defined two vectors—principal stress1 and principal stress2. Unlike line integral convolution, a streamline must be computed at every node to act as a filter kernel and convolve this filter kernel with white noise. We also compute Perlin Noise at every node and convolve it with a simple triangle filter to obtain the opacity value for each pixel of the image. Then color is used to encode the eigenvalue of each principal stress. Finally, the color and opacity are composed together to get the final images (see Figures 7 and 8). A scale from saturated blue(negative) via grey to saturated red(positive) is used (see Figure 17).

The above process can be described using equation 4 and equation 5:

$$O(x,y) = \int_{-\infty}^{\infty} \mathrm{kernel}(u)\mathrm{PerlinNoise}(x-u, y-u)du \tag{4}$$

$$O(x,y) = \int_{-\infty}^{\infty} \mathrm{kernel}(u)$$
$$\mathrm{PerlinNoise}(x-u\sin\alpha(x,y), y-u\cos\alpha(x,y))du \tag{5}$$

Where equation 4 is a simple convolution, and equation 5 rotates the parameter of Perlin Noise along the vector direction, $\alpha(x,y)$ is the angle between the x axis and the first or second principal stress vector.

4.1. Evaluation

All the texture images' resolution here are 1024×1024. Figures 7 and 8 show two minimum and maximum principal stress trajectories respectively. In Figures 9 and 10 , the Perlin Noise used to visualize the two stress vector fields has 10 times lower frequency than those in Figures 7 and 8. Recall the section 2, in Figure 2, lower frequency correspond to fractal like texture. If users increase the scale, the generated image is Figure 11, which is the visualization of the same stress fields, but except for color, there shows no pattern at all. Its frequency is 10 times higher than those in Figures 7 and 8 so that generated image is similar to white noise. Figures 12 and 13 compare to Figures 9 and 10 show obvious artifacts due to high frequency resulting from bigger parameters. Figures 14 and 15 compare to Figures 7 and 8 show obvious artifact due to high frequency resulting from bigger parameters as well.

From these images, we can see clearly that the two principal stresses are almost normal to each other. And two obvious advantages of using texture lines are that it depicts

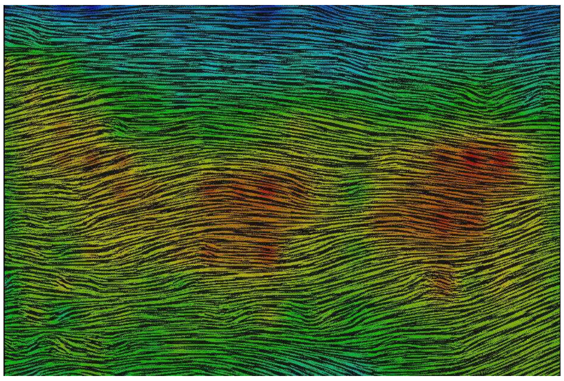

Figure 7: *Texture visualization of minmum principal stress fields*

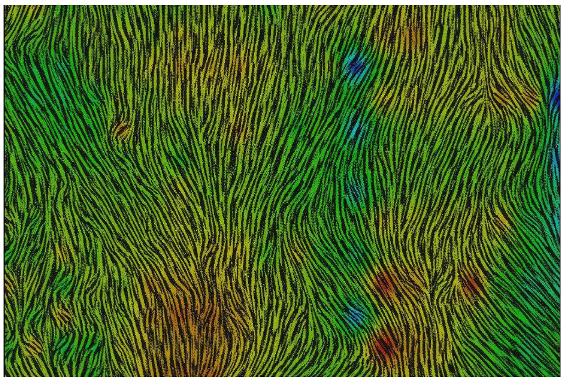

Figure 8: *Texture visualization of maximum principal stress fields*

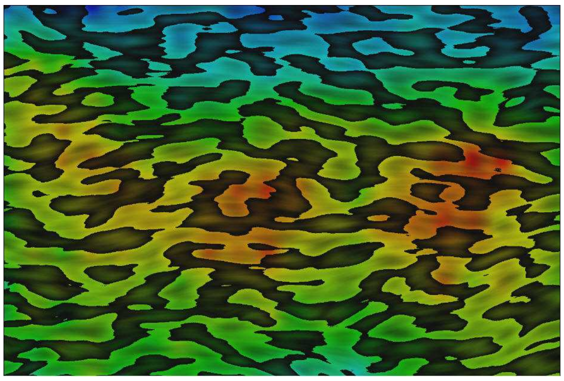

Figure 9: *The minumum principal stress trajectory visualization correspond to texture in Figure 2*

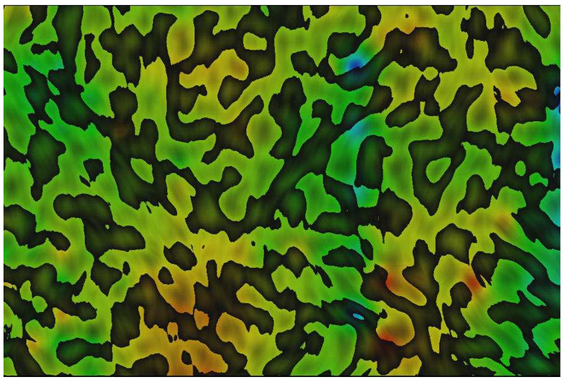

Figure 10: *The maximum principal stress trajectory visualization correspond to texture in Figure 2*

the tnesor field continuously, and it is space filling so that a better understanding of the whole stress tensor fields structure can be achieved. Furthermore, the user can control the output visualization simply by modifying the parameters of a Perlin Noise.

5. Conclusions and Future Work

We have evaluated Perlin Noise application for a 2D rail head stress data set. The resulting images provide a good visual cue of the whole stress tensor field for a rail head. Obviously, besides LIC and Spot Noise, Perlin Noise is another effective texture visualization method. Compare to LIC, Perlin Noise doesn't need to do the integration to get the streamline, which reduces the complicity of computation. Spot Noise is essentially the same as the Perlin Noise, except that spot shape is based on data at a single point. If the vector varies strongly over this region, the shape of spot does not reflect the data properly [de 97]. Perlin Noise is as good as LIC in capturing the high vector gradient in a region. This is at present a work-in-progress paper and further comparison of noise is still required, as well as the use of the different noise types.

The rail stress tensor field used is stationary, so that there is no need to track the change of the tensor field. However, if the tensor field is time dependent, how to animate the Perlin Noise texture fast and track the changes of the eigenvector field will be a main concern. And because measured data are quite uniform, we use a unifrom grid to generate texture. How to deal with a non-uniform curvilinear grid will be one of our concerns. Another important issue is that engineers are particular interested in cracks in the residual stress field. For a rail head, a lot of important stress feature are found near the edge areas, therefore, the resulting texture near edges will need to be explored in more detail.

6. Acknowledgments

This work is funded by the Computer Science School of The University of Manchester.

The authors greatly appreciate J. Kelleher who provided

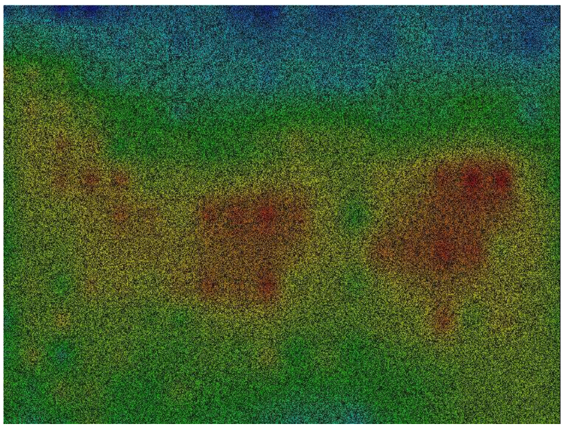

Figure 11: *Cannot see any direction pattern, more white noise like result*

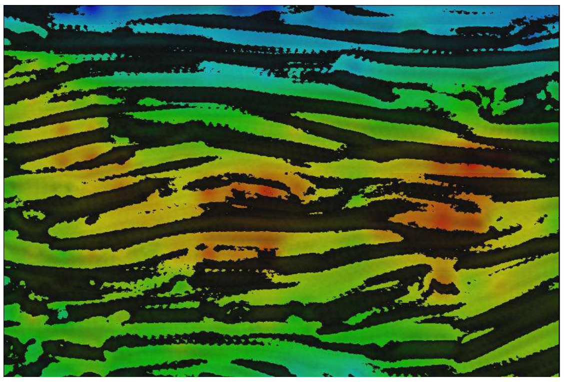

Figure 12: *Texture visualization of minmum principal stress fields with obvious artifact*

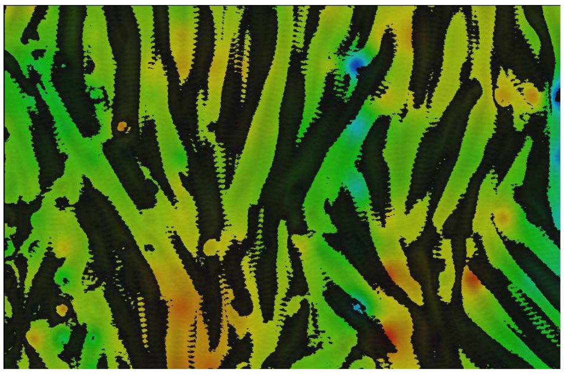

Figure 13: *Texture visualization of maximum principal stress fields with obvious artifact*

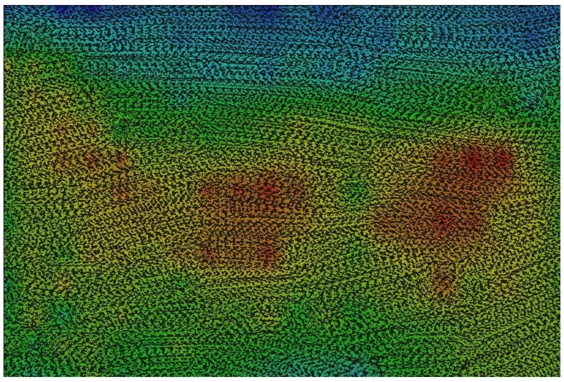

Figure 14: *Texture visualization of minmum principal stress fields with obvious artifact*

the rail tensor data and a lot of insightful comments on stress tensors. And thank for Tobias Schiebeck, Yien Kwok, Mary McDerby, and all other Manchester Visualization Center students and staffs for their help and support during this project.

References

[CL93] CABRAL, LEEDOM L.: Image vector fields using line integral convolution. *Computer Graphics 27* (1993), 263–272.

[de 97] DE LEEUW W.: *Presentation and exploration of flow data*. PhD thesis, Technology University of Delft, 1997.

[Del94] DELMARCELLE T.: *The visualization of second-order tensor fields*. PhD thesis, Stanford University, 1994.

[Hab90] HABER R.: Visualization techniques for engineering mechanics. *Computing Systems in Engineering 1* (1990), 37–50.

[LHD*04] LARAMEE R., HAUSER H., DOLEISCH H., VROLIJK B., POST F., WEISKOPF D.: The state of the art in flow visualization: dense and texture-based techniques. *Computer Graphics Forum 23*, 2 (2004), 203–221.

[Per85] PERLIN K.: An image synthesizer. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (1985), vol. 19, pp. 287–296.

[PH89] PERLIN K., HOFFERT E. M.: Hypertexture. *Computer Graphics 23*, 3 (July 1989), 253–261.

[PTVF02] PRESS W., TEUKOLSKY S., VETTERLING W., FLANNERY B.: *Numerical Recipes in C++: The Art of Scientific Computing*, second ed. Cambridge University Press, 2002.

[UTFK*89] UPSON C., THOMAS FAULHABER J., KAMINS D., LAIDLAW D. H., SCHLEGEL D., VROOM J., GURWITZ R., VAN DAM A.: The application visualization system: A computational environment for scientific visualization. *IEEE Comput. Graph. Appl. 9*, 4 (1989), 30–42.

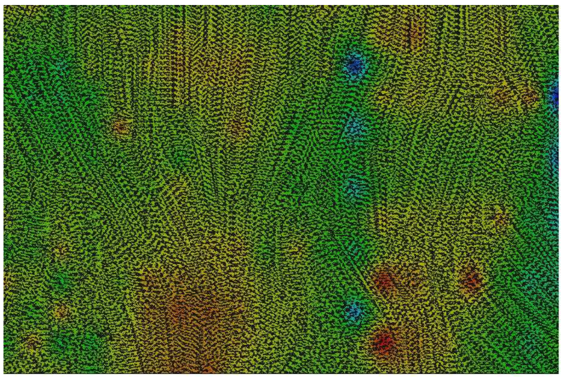

Figure 15: *Texture visualization of maximum principal stress fields with obvious artifact*

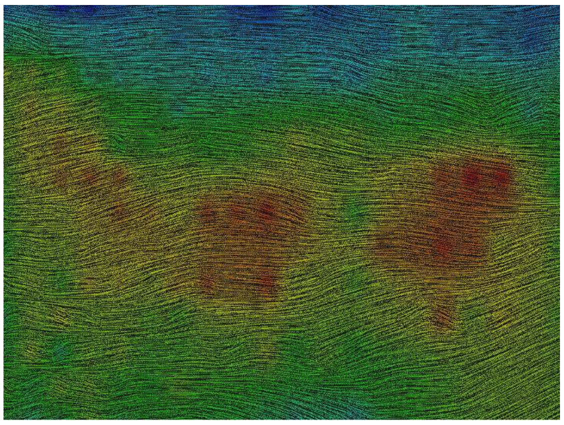

Figure 16: *Better result of texture visualization of stress trajectory*

[Van91] Van Wijk J.: Spot noise: Texture synthesis for data visualization. *Computer Graphics 25*, 4 (July 1991), 309–318.

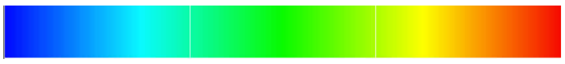

Figure 17: *The colour scale*