A speckle-texture image generator

Jean-José Orteu, Dorian Garcia, Laurent Robert, Florian Bugarin

A Speckle-Texture Image Generator

Jean-José Orteu and Dorian Garcia and Laurent Robert and Florian Bugarin
École des Mines d'Albi, Campus Jarlard, F-81013 Albi CT Cedex 09, FRANCE

ABSTRACT

We propose a framework for obtaining synthetic speckle-pattern images based on successive transformations of Perlin's coherent noise function. In addition we show how a given displacement function can be used to produce deformed images, making this framework suitable for performance analysis of speckle-based displacement/strain measurement techniques, such as Digital Image Correlation, widely used in experimental mechanics.

Keywords: synthetic speckle-pattern images, digital image correlation, displacement accuracy assessment

1. INTRODUCTION

A major challenge posed by speckle analysis techniques for metrology applications, such as Digital Image Correlation (DIC), is their performance assessment. As the number of influencing parameters is large, a practical approach consists in analysing computer generated images given sets of varying settings. Several authors[1–4] have used synthetic deformed speckle-pattern images for DIC evaluation, however the generated patterns are quite non realistic, often exhibit very different spectral properties as present in real speckle images, and elude potentially important effects brought by the digitization process of an imaging system.
We want to produce synthetic images of a realistic speckle pattern. Among many different approaches discussed in,[5] the one we propose is based on a coherent noise generator and a set of continuous transformations to produce the desired pattern aspect.

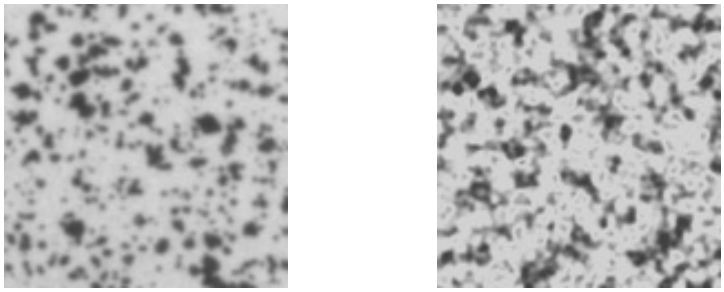

Figure 1. Real pattern obtained by spray-paint (left) and synthetic pattern realized with our coherent noise generator (right).

Our objectives are: (a) to mimic speckle patterns which would be obtained using regular approaches such as spray painting, toner powder deposit, ...; (b) to simulate the sensor fill-factor to exhibit eventual moiré structures due to the spatial sampling characteristics; (c) to limit the introduction of a bias of some sort due to interpolation; (d) to be deterministic: running twice the algorithm with the same parameters must produce the same pattern; (e) to simulate a deformation field of arbitrary type; (f) to simulate lens distortion of arbitrary type if necessary; (g) to be approved by the photomecanic community to standardize performance evaluations.

2. GENERATION OF THE SPECKLE IMAGE

The core speckle pattern generator algorithm is presented in Figure 2 and consists of five elementary steps. The algorithm is given at the end of the paper.

Further author information: E-mail: Jean-Jose.Orteu@enstimac.fr, Telephone: +33 (0)563 493 073

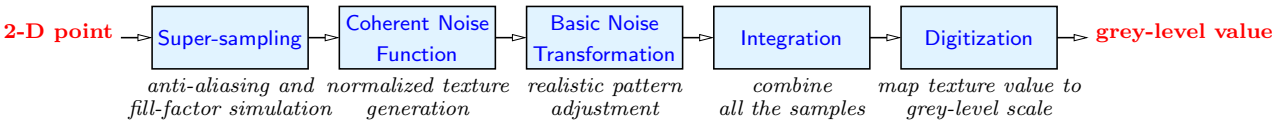

Figure 2. Processing pipeline for computing the pixel value within a speckle-image given its coordinates.

2.1. Super-sampling and fill-factor simulation

The integration of the texture function over the domain corresponding to the sensitive photometric material of a pixel is performed by a sub-sampling technique. To reduce the band-limitation of the texture spectrum that a regular grid-sampling technique would produce, we perform a Monte-Carlo integration scheme using stochastic samples to approximate the integral. Stochastic sampling scatters aliasing into noise, thus eliminating any systematic bias in the generated patterns if the sample distribution is chosen carefully. We have opted for the Jittering distribution which is both computationally efficient and has satisfactory spectral properties.

Given a two-dimensional point (u, v), a set of super-sampled coordinates (u_i, v_i) (i.e. set of neighboring points) is generated (see Figure 3-left). In modern CCD cameras it is possible that a portion of the camera surface is not sensitive to light (see Figure 3-middle). The fraction of the surface that is sensitive to light is called the fill-factor. In our algorithm, a fill-factor can be taken into account during the super-sampling process (see Figure 3-right).

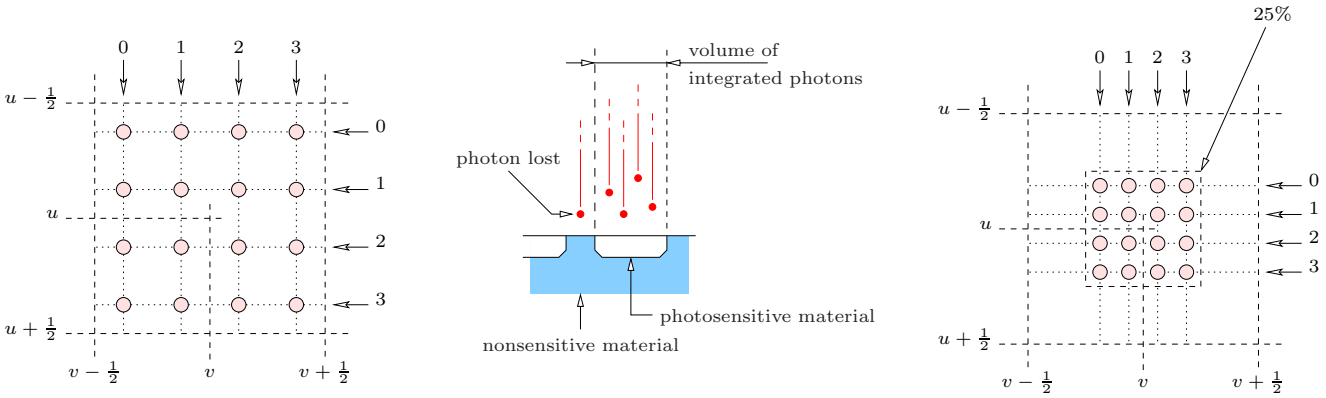

Figure 3. Super-sampling (left) ; fill-factor (middle) ; super-sampling with fill-factor (right).

In reality, as mentioned above, we have implemented a Jittered distribution by slightly perturbing each samples of the regular grid in a random direction. The super-sampled coordinates are generated using the following equations:

$$\begin{aligned} u_i &= u + \sqrt{f}\,\frac{2i+1-n}{2n} + rd \\ v_j &= v + \sqrt{f}\,\frac{2j+1-n}{2n} + rd \end{aligned} \tag{1}$$

where rd is used for jittering the super-sampled coordinates. The random variable rd is uniformly distributed over a range which is proportional to the super-sampling spacing $\sqrt{f}/n$, and further controlled by a jittering rate noted J: $rd = J\,\text{random}\,([-1, +1])\,\sqrt{f}/n$. Hence, rd is uniformly distributed over the range $\pm J\,\sqrt{f}/n$.
It is important to note that in order for the algorithm to be deterministic, the random function must be approximated by a pseudo-random generator as found readily available on most software platforms.

2.2. Coherent Noise Function

2.2.1. Basic Noise Function

The basic algorithm to generate a coherent noise is to setup a regular grid of random values and then interpolate these values at non lattice positions. Perlin[6] first proposed to generate random vectors n_i at lattice points m_i, then interpolate these vectors by the means of a dot-product $s_i = \mathbf{n}_i \times \mathbf{d}_i$, $i = 1 \dots 4$ (see Figure 4-left) and a s-shaped weight function in order that the noise function be more influenced by a coefficient s_i when the point m is near from m_i.
Figure 4-right shows a coherent noise generated using Perlin's algorithm.[6,7] As it can be seen in that figure, Perlin's noise function is inherently continuous and is made up with "bumps", or grains, of roughly the same size.

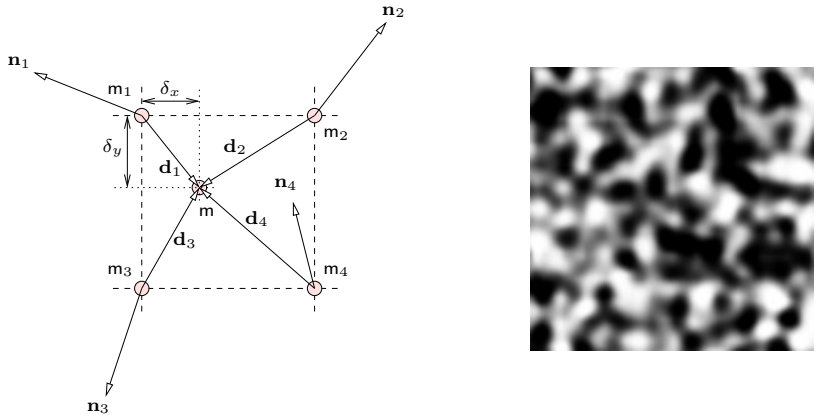

Figure 4. Generation of the coherent noise (left) and corresponding speckle-pattern image (right).

2.2.2. "Harshness" Control

The noise function as described in the previous section does produce only one "grain size" as shown in Figure 4-right. This characteristic is defined as the wavelength of the noise signal: the distance separating two lattice points. The amplitude of the noise signal is defined as the difference between the maximum and the minimum values the noise function can be.
Some speckle patterns are sometimes made up with many different grain sizes (spray painted speckle is one example). We can have control over it by summing noises of different wavelengths (frequencies **F**) and amplitudes **A**. In 1989, Lewis[8] proposed two new interpolation schemes to solve Perlin's algorithm issues. Both of them offers spectral control over the generated noise. Results obtained with one of his algorithm are shown in Figure 5.

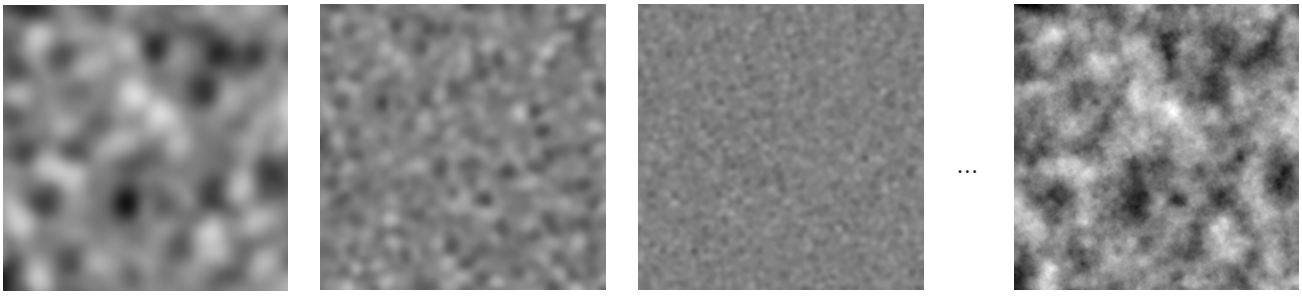

Figure 5. Controlling the noise "harshness": the bottom-right coherent noise is built by summation of five basic noises of different frequency **F**=1,2,4,8,16 and amplitude **A**=1,1/2,1/4,1/8,1/16 (Lewis' sparse convolution algorithm is used for the basic noise function).

2.3. Basic Noise Transformation

The raw coherent noise function represents a single type of speckle pattern which may not be appropriate for all applications. Especially, the noise function has a rather uniform distribution hence it is not appropriate for tests requiring bi-modal distributions for instance. It is therefore possible to apply a transformation to this coherent noise function so that the transformed outputs mimic one desired speckle pattern appearance. Figure 6 shows some patterns which can be realized through the use of some specific, very simple, texturing functions.
More effects can be achieved by modulating the texturing function with respect to the point location, eventually corrupting the point location by an other noise function. Such a technique can allow a better control of the speckle spots by using, for instance, a sine function with a corrupted phase. Pixar* has successfully implemented this approach in its Renderman software for producing leopard texture, marble, etc.

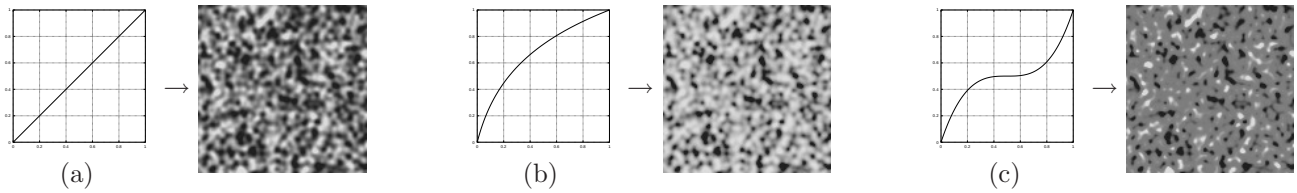

Figure 6. Examples of some simple functions which transform a coherent noise function into some more realistic speckle patterns. (a) $\mathrm{T}(x) = x$; (b) $\mathrm{T}(x) = \log(9\,x + 1)$; (c) $\mathrm{T}(x) = 4\left(x - \frac{1}{2}\right)^3 + \frac{1}{2}$

3. DEFORMATION OF THE SPECKLE IMAGE

Knowing how to generate a synthetic speckle-pattern image, it is now straightforward to use it for simulating a motion or, in general, a deformation. The principle consists of determining the inverse of the displacement function and evaluate the texture pattern at each inverse mapped point (see the algorithm at the end of the paper). Let's say the inverse of the displacement function is:

$$\begin{aligned} \mathbf{d}^{-1} : \mathbb{R}^2 &\longrightarrow \mathbb{R}^2 \\ (x, y) &\longmapsto \mathbf{d}^{-1}(x, y) \end{aligned}$$

The first pattern is generated through the $noise(\cdot)$ function following the algorithm described in the previous section:

$$(x, y) \longmapsto noise(x, y)$$

The deformed pattern is then computed using the inverted displacement function $\mathbf{d}^{-1}(\cdot)$:

$$(x, y) \longmapsto noise\left(\mathbf{d}^{-1}(x, y)\right)$$

The inverse of the displacement function is not always easily known (if possible). However it can be approximated numerically or one can setup directly the inverse displacement as a well-defined function. For example, using a "punch" function:[5]

$$\mathbf{d}^{-1}(x, y) = \frac{1 + p(x^2 + y^2)}{1 + pr^2} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

where x' and y' are coordinates normalized in the range $[-1 \cdots 1]$, p is the power of the deformation (maximum magnitude of the gradient), and r defines the radius of a circle at which no point is deformed (the function is invariant at the origin and on this circle).
The deformation that are implemented so far in the software are: the scale (to control the speckle-pattern grain size), a translation, a rotation, a punch (for demonstration), a sinusoidal displacement (see section 4), but any displacement function could be implemented. Figure 7 illustrates the punch function.

*Pixar is the animation company which realized "Toys Story", "A Bug's Life", "Toys Story II", "Monsters, Inc", and "Finding Nemo". `http://www.pixar.com`

$p = 0$ $p = 1$

Figure 7. Punch function. We show two results obtained with different values of the parameter p of the punch function; $r = 0.8$. The meshes give a better idea of how the original pattern is deformed by tracking the displacements of the points on a regular grid.

4. APPLICATIONS

Digital Image Correlation (DIC) is a widely used technique for full-field displacement/strain measurements in experimental mechanics.[9] From a metrology point of view, it is crucial to assess the performance of the technique and to determine the influence of the tuning parameters on the accuracy of the measurements. The French working group "GdR CNRS 2519"** has started a study to assess the performances of the DIC technique for displacement/strain measurements.[10] For this study, images with a speckle-pattern and submitted to a sinusoidal displacement at various frequencies and amplitudes have been generated. The displacements evaluated using DIC are compared with the exact imposed displacement values. Deformed images (image size 1024×1024) are obtained assuming a plane wave sinusoidal displacement† with only tension/compression waves in the X direction (horizontal direction) and a zero displacement along Y (see Figure 8): $u_X(X,Y) = \alpha\, p \sin\left(2\pi \dfrac{X}{p}\right)$, where p is the period in pixels and α is the amplitude of the XX component of the displacement gradient (strain).

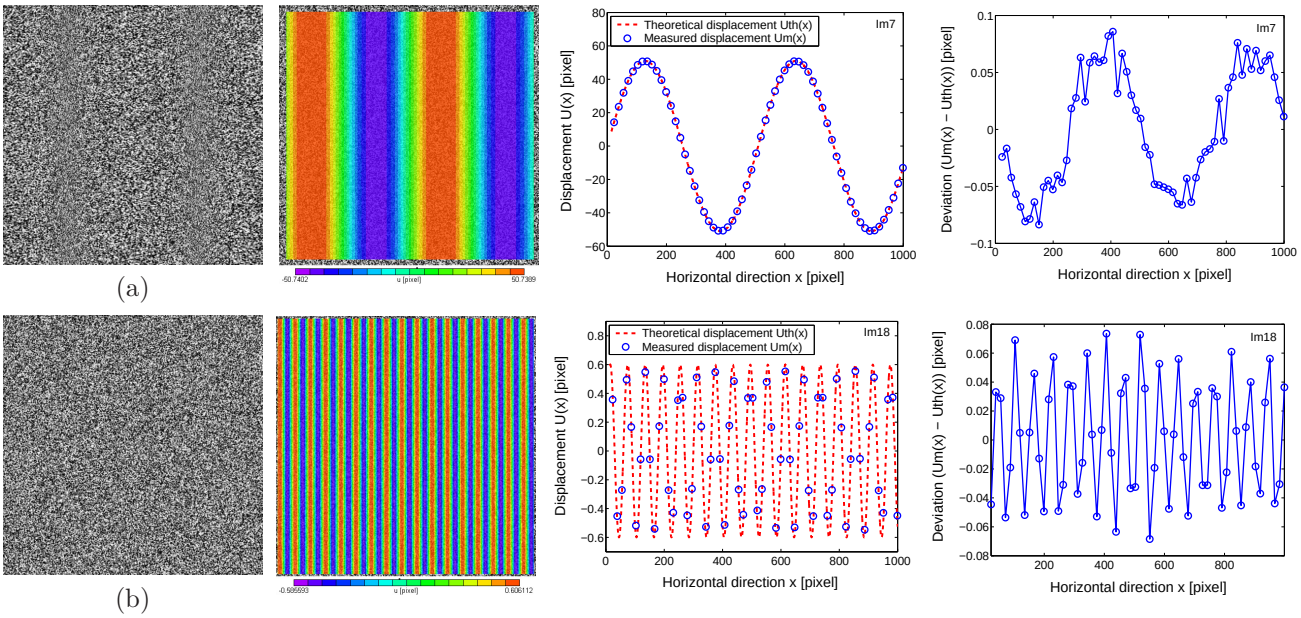

Figure 8. From left to right: deformed image, displacement field computed using digital image correlation, computed displacement vs real imposed displacement (along an horizontal line of the image), deviation between computed and real displacements (along an horizontal line of the image) – Two situations are shown: (a) $\alpha = 0.1$, $p = 510$ (displ. max = 51 pixels, def. max = 63%) – (b) $\alpha = 0.01$, $p = 60$ (displ. max = 0.6 pixels, def. max = 6.3%).

**http://www.ifma.fr/lami/gdr2519/

†As mentioned in section 3, the inverse of the displacement function is required to generate the deformed image. To compute a sinusoidal displacement at X, we have to find X_0 such that: $X = X_0 + u_X(X_0)$. In order to find X_0, we solve $g(X_0) = X_0 + u_X(X_0) - X = 0$ using the Newton algorithm.

5. CONCLUSION

We have shown how Perlin's noise function can be used in a flexible speckle generation framework suitable for performance assessment of various measurement techniques, such as digital image correlation (DIC).
The multi-step approach used within this framework allows one to extend the range of simulated images by introducing a new transformation in the processing pipeline. We have given an application example of such an extension for simulating a displacement function, and used the results for the preliminary performance assessment of a DIC algorithm.

Algorithm 1 Generate a (deformed) speckle-pattern image

for each pixel (u, v) **do**
  // super-sampling ($n \times n$ neighbourhood) and fill-factor
  sumNoise $\leftarrow 0$
  for (i, j) in $1 \cdots n \times 1 \cdots n$ **do**
    // sample coordinates
    $(u_i, v_j) \leftarrow function(u, v, i, j, n, f, J)$ *// see equation (1)*
    if deformation **then**
      $(u_i, v_j) \leftarrow d^{-1}(u_i, v_j)$ *// deformation (optional)*
    end if
    sumNoise $\leftarrow$ sumNoise + coherentNoise(u_i, v_j) *// sample evaluation and aggregation*
  end for
  noise $\leftarrow$ sumNoise$/n^2$ *// average*
  noise $\leftarrow$ T(noise) *// pattern adjustment, see Figure 6*
  // photometric mapping and digitization
  photometry $\leftarrow$ noise $\times$ (maxGreyLevel - minGreyLevel) + minGreyLevel
  pixel$(u, v) \leftarrow$ round(photometry)
end for

REFERENCES

1. B. Wattrisse, *Étude cinématique des phénomènes de localisation dans des aciers par intercorrélation d'image.* PhD thesis, Université de Montpellier II (France), Feb. 1999.
2. P. Doumalin, *Microextensométrie locale par corrélation d'images numériques. Application aux études micromécaniques par microscopie électronique à balayage.* PhD thesis, École Polytechnique (France), June 2001.
3. H. W. Schreier and M. A. Sutton, "Systematic Errors in Digital Image Correlation Due to Undermatched Subset Shape Functions," *Experimental Mechanics* **43**(3), pp. 303–311, 2002.
4. M. Stanislas, K. Okamoto, C. J. Kahler, and J. Westerweel, "Main results of the Second International PIV Challenge," *Experiments in Fluids* **39**, pp. 170–191, 2005.
5. D. Garcia, *Mesure de formes et de champs de déplacements tridimensionnels par stéréo-corrélation d'images.* PhD thesis, Institut National Polytechnique de Toulouse (France), Dec. 2001.
6. K. Perlin, "An image synthesizer," in *SIGGRAPH'85*, pp. 287–296, 1985.
7. K. Perlin, "Making noise." http://www.noisemachine.com/talk1, 1999. Presentation based on a talk presented at GDCHardCore.
8. J. P. Lewis, "Algorithms for solid noise synthesis," *Computer Graphics* **23**(3), pp. 263–270, 1989.
9. T. C. Chu, W. F. Ranson, M. A. Sutton, and W. H. Peters, "Applications of Digital-Image-Correlation Techniques to Experimental Mechanics," *Experimental Mechanics* **25**(3), pp. 232–244, 1985.
10. M. Bornert, "Resolution and spatial resolution of digital image correlation techniques," in *Photomechanics'2006*, Clermont-Ferrand (France), 10-12 July 2006.