IOP Publishing

Leonid P Yaroslavsky

Chapter 6

Statistical image models and pattern formation

6.1 Introduction

Statistical image models are mathematical models for generating pseudo-random images with certain prescribed statistical properties. They are used in simulations of imaging systems and testing image processing algorithms, as well as for the synthesis of artificial images that imitate natural ones. The exercises in this chapter demonstrate an algorithmic approach to building statistical image models, according to which the models are built by means of certain (specific for each model) combinations of certain standard elementary operations applied in a certain order to a starting, or seed, image. Implemented here are the following models listed in the entrance menu (figure 6.1):

- *Point-wise nonlinearity (PWN) model*: seed images are subjected, pixel by pixel, to a point-wise nonlinear transformation specified by its amplitude transfer function.
- *Linear filter (LF) model*: seed images are subjected to a linear filtering with a certain prescribed filter point spread function or frequency response.
- *Point-wise nonlinearity and linear filter (PWN&LF) model*: assumes point-wise nonlinear transformation and linear filter in cascade, i.e. seed images are first subjected to a point-wise nonlinear transformation and then the result is fed to a linear filtering.
- *Linear filter and point-wise nonlinearity (LF&PWN) model*: assumes linear filtering and point-wise nonlinear transformation in cascade in the opposite order: seed images are first subjected to a linear filtering and then to a point-wise non-linear transformation.
- *Evolutionary models*: iterative, i.e. models with a feedback: at first iteration a seed image is subjected to a certain transformation specific for a particular model and then at other iterations, this transformation is applied to the image obtained at the previous iteration.

doi:10.1088/978-0-7503-1233-2ch6

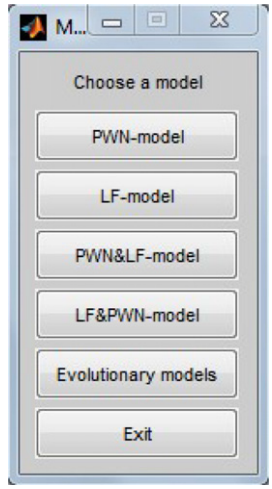

Figure 6.1. Statistical image models and pattern formation: start menu.

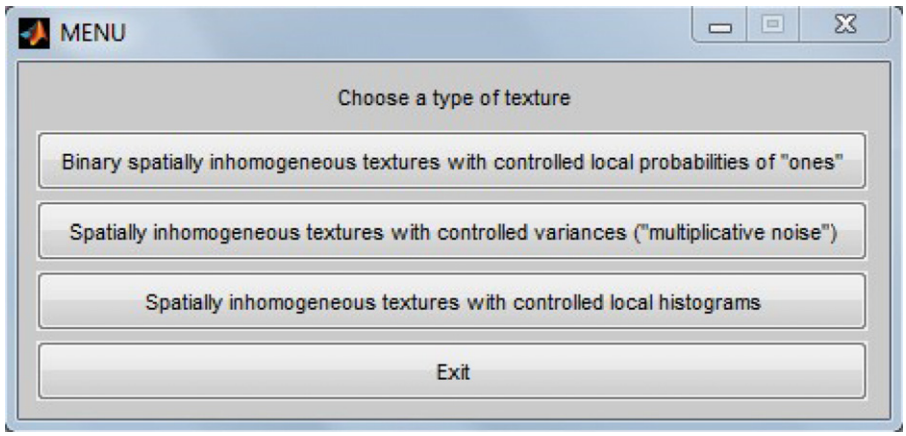

Figure 6.2. PWN-models: start menu.

6.2 PWN models

PWN models are the simplest ones in the family of the image statistical models and pattern generators. Using point-wise transformation, one can control probability histograms of output images, and this is the only image statistical parameter that can be directly controlled in this model. Three versions of PWN models implemented for exercises are listed in the menu shown in figure 6.2.

6.2.1 Binary spatially inhomogeneous texture with controlled local probabilities of 'one'

In this exercise one can generate binary pseudo-random images with probability of 'one' proportional, for each pixel, to normalized to the unity gray level of its

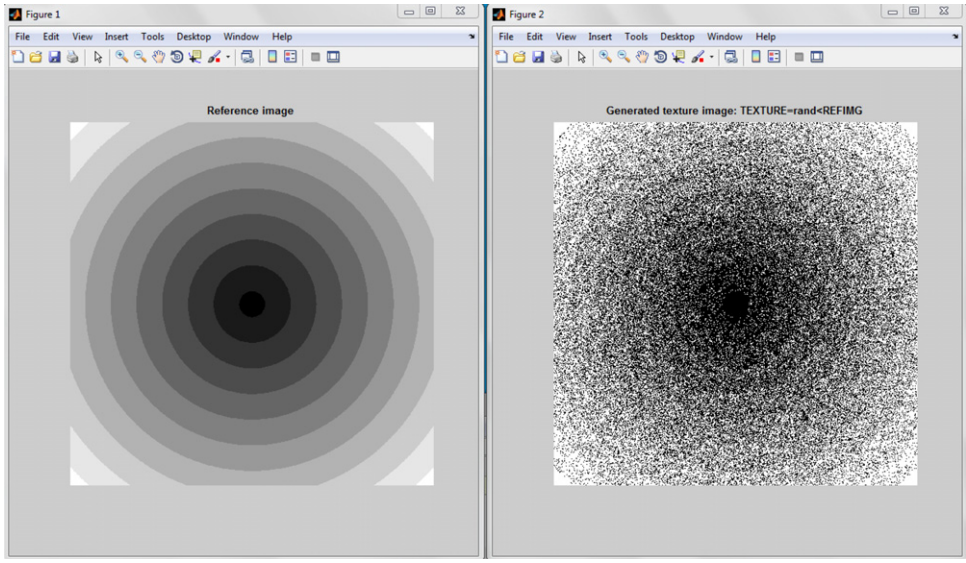

Figure 6.3. PWN model: reference image (left panel) and an example of a binary pseudo-random image with density of 'ones' (shown in white) controlled by the reference image (right panel).

corresponding, i.e. having the same coordinate, pixel of a reference image. The resulting images are, naturally, spatially inhomogeneous according to the spatial inhomogeneity of the reference images. As seed images, arrays of uncorrelated pseudo-random numbers with uniform distribution obtained by means of the standard MATLAB® generator of 'random' numbers are used. The nonlinearity implemented in the model is a threshold nonlinearity, the threshold being equal, for each particular pixel, to the normalized gray levels of the corresponding pixel of the reference image. Reference images are to be chosen by the user from the image database. An example of a pattern generated by the model is shown in figure 6.3.

6.2.2 Spatially inhomogeneous texture with controlled variances ('multiplicative noise')

In this version of the PWN model, generated are zero mean pseudo-random images with normal distribution and standard deviation proportional, for each pixel, to the normalized gray level of the corresponding pixel of a reference image. This type of pseudo-random image simulates signal dependent multiplicative noise. As seed images, pseudo-random numbers with normal distribution taken from the standard MATLAB® pseudo-random number generator are used. The amplitude transfer function of the 'nonlinearity' is a parabolic transfer function, which spans within the image dynamic range ([0–255]) with a slope proportional to its argument, the gray level. As in the previous exercise, reference images are to be chosen by the user. Figure 6.4 presents an example of an image generated in this way and its corresponding reference image.

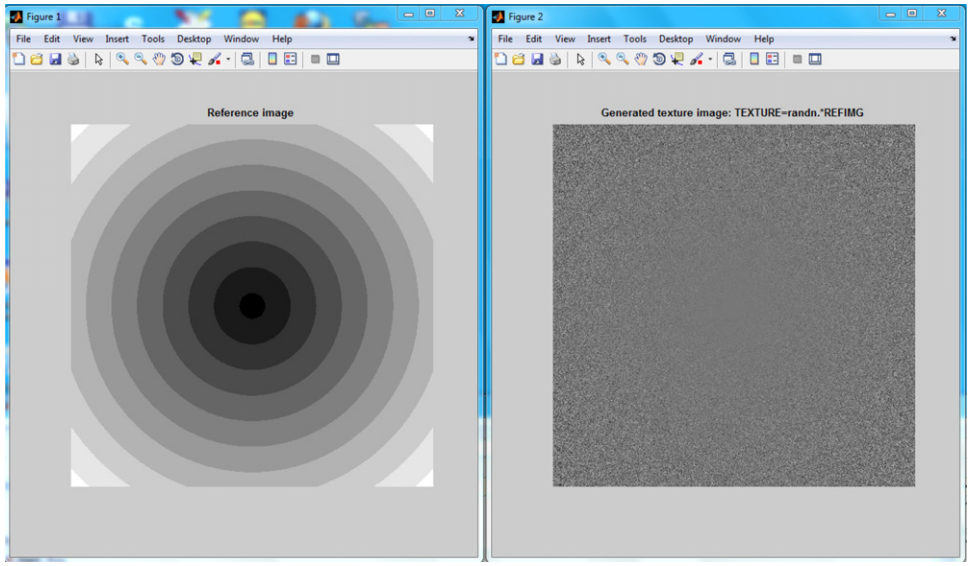

Figure 6.4. PWN model: a reference image (left panel) and an example of a pseudo-random image with pixel gray level standard deviations proportional to gray levels of corresponding pixels of the reference image (right panel).

6.2.3 Spatially inhomogeneous texture with controlled local histograms

This implementation of the model enables generation of pseudo-random images with local histograms within windows of a given size identical to local histograms of a reference image in corresponding positions of windows. As seed images, arrays of uniformly distributed pseudo-random numbers generated by the standard MATLAB® random number generator are used. The model nonlinearity is the one that converts uniform distribution of pseudo-random numbers into a given distribution. It is inverse to the *histogram equalization* transformation. The reference images as well as vertical and horizontal dimensions of the window for computing local histograms are to be chosen by the user. A result of the work of the model is illustrated in figure 6.5.

6.3 LF models

6.3.1 Introduction

While PWN models control image gray level distribution histograms, linear filter (LF) models generate images with pre-defined power spectra (or, correspondingly, autocorrelation functions). It is noteworthy that, by virtue of the central limit theorem of the probability theory, gray level distributions of images generated by LF models tend to a normal (Gaussian) distribution.

Exercises in this section demonstrate synthesized images for several types of image spectrum and a possibility of generating images that imitate certain types of natural texture. Linear filters in the models are implemented in spectral DFT or DCT domains: seed arrays of pseudo-random uncorrelated zero mean numbers with

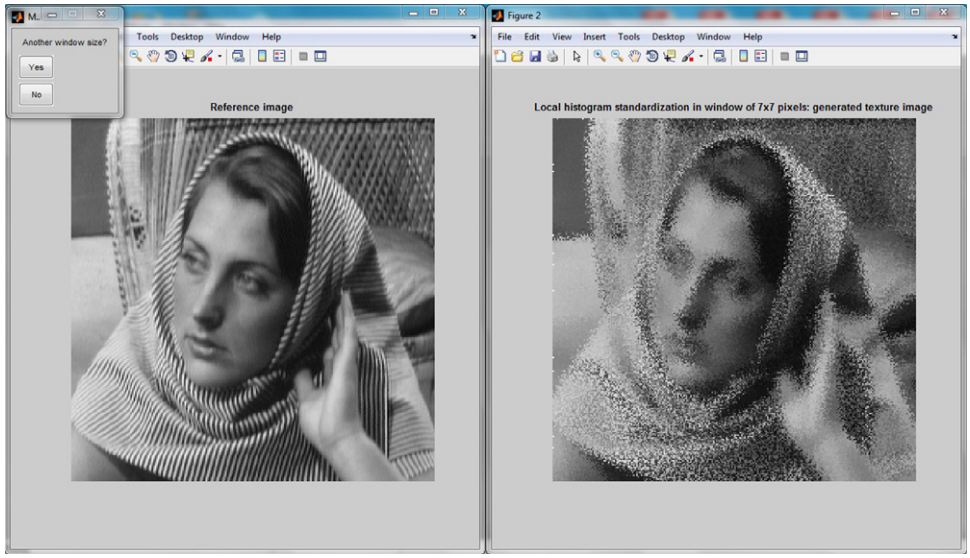

Figure 6.5. PWN model: a reference image (right) an example of a pseudo-random image with local histograms over a certain window identical to local histograms of the reference image in corresponding 7 × 7 pixel window positions.

uniform distribution in the range [−0.5, 0.5]) from the standard MATLAB® random number generator are multiplied by spectral masks of certain chosen by the user forms or controlled by a reference image, and the images are obtained by inverse DFT or DCT transform. Thus, power spectra of generated images are specified by the spectral masks, while spectra phase components are determined by the pseudo-random numbers. Types of texture images that can be generated are listed in the start menu in figure 6.6.

6.3.2 'Ring of stars', circular and ring-shaped spectra, 'fractal' textures

'Ring of stars' textures are generated in the DFT or in DCT domains using spectral masks in the form of a 'ring of stars' (several spectral components uniformly arranged over a circumference of a circle (for DFT) or of a pie-sector (for DCT) of certain radius centered at the DC component). For generating circular and ring-shaped spectra textures, spectral masks in the form of circles and rings, for DFT, or pie-sectors and quarters of rings, for DCT, are used. For generating 'fractal' textures, one uses spectral masks in the form of the DC component centered at the spectrum circularly symmetrical surfaces that decay inversely proportionally to the 0.5th; 1st; 1.5th; and 2nd powers of the distance from the DC component ('$1/f^P$'-type spectra).

For generating 'ring of stars' textures, the user is prompted to choose DFT or DCT transform domains for filtering, set the radius of a circle as a fraction of the image size, and the number of spectral components ('stars') uniformly arranged on the circumference of the circle or, correspondingly, pie-sector. For textures with

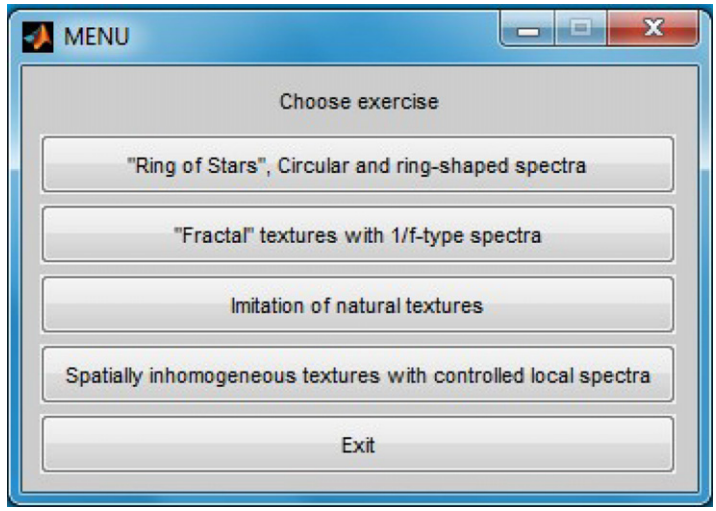

Figure 6.6. LF models: start menu.

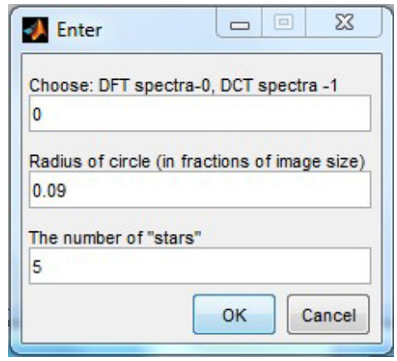

Figure 6.7. LF model, 'ring of stars', circular and ring-shaped spectra: menu of parameters.

circular or ring-shaped spectra, the user defined parameters are corresponding radii (figure 6.7).

The program can be run many times producing different realizations of this type of textures with the same parameters. Examples of the resulting images are presented in figure 6.8.

If the option 'fractal' textures with 1/f-type spectra is chosen, the program automatically generates all four special cases of fractal textures with $1/|f|^P$ spectra for $P = 0.5$, $P = 1$, $P = 1.5$ and $P = 2$ (figure 6.9).

6.3.3 Imitation of natural textures

A remarkable property of LF models is their capability of imitating many types of natural texture. This can be done by assigning to arrays of pseudo-random images the power spectrum of a template texture images to be imitated. To demonstrate this

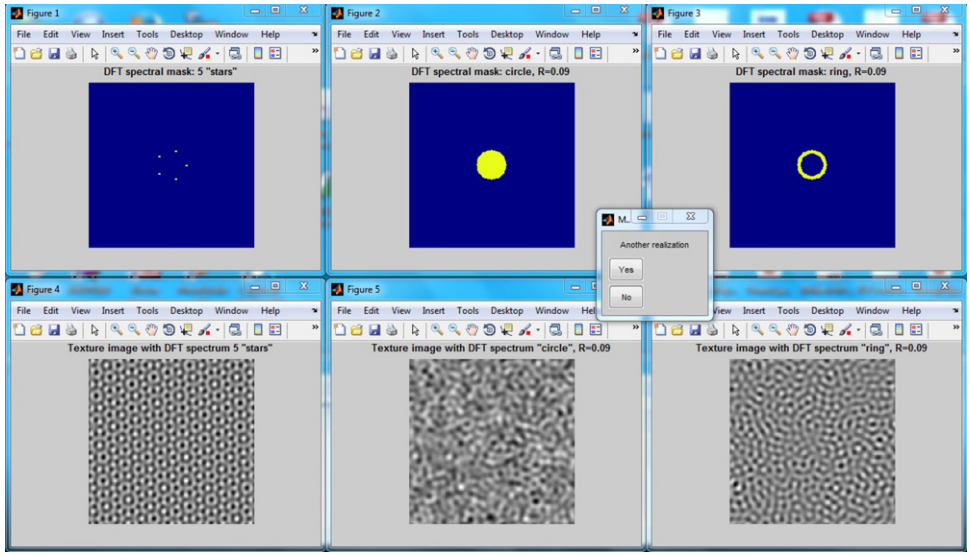

Figure 6.8. LF models, 'ring of stars', circular and ring-shaped spectra. Shown are examples of generated images (bottom row) along with corresponding frequency responses of linear filters of the models. (Yellow color in images of spectral masks corresponds to ones; blue color corresponds to zeros in the frequency responses).

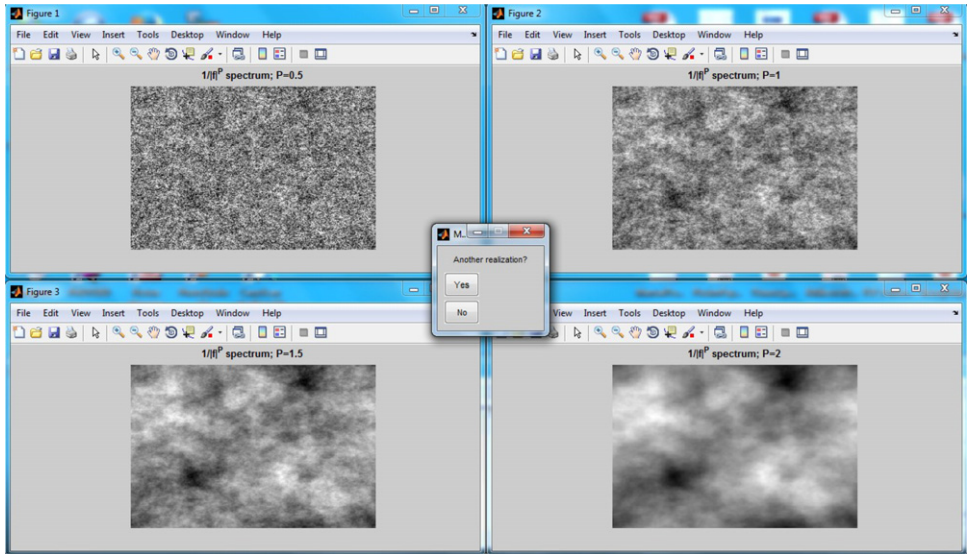

Figure 6.9. LF models, 'fractal' textures with circular symmetric power spectra that decay inversely proportionally to powers 0.5, 1, 1.5 and 2 of the spatial frequency.

capability, two options are offered in the exercise for choosing the template texture: (i) three types of natural textures, a 'wood' texture of wooden furniture, a 'mohair' texture of weaved woolen fabric and another texture, 'textile', of woven fabric, and (ii) arbitrary texture image from the image database. The latter option is intended

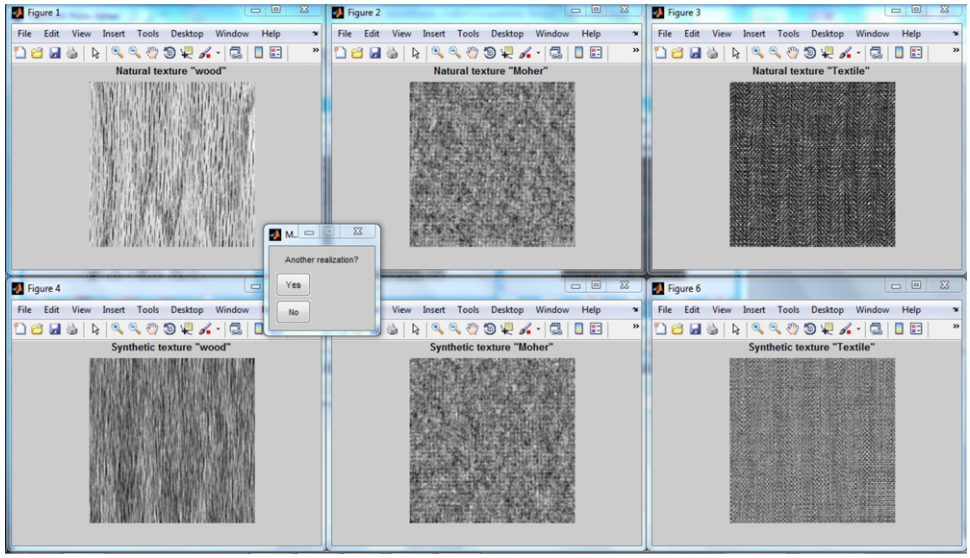

Figure 6.10. LF models: samples of natural textures (upper row) and their computer generated imitations (lower row).

for demonstration that not all natural texture can be imitated using the LF model. The program can be run many times to generate different realizations of texture images of the same type, which, in this case, is defined by the power spectrum of template images. Examples of generated texture images that imitate natural prototypes are shown in figure 6.10.

6.3.4 Spatially inhomogeneous textures with controlled local spectra

In the previous exercise, LF models are applied globally, i.e. to spectra of whole images. Resulting textures are, therefore, spatially homogeneous (in terms of their local spectra). Local application of LF models enables generating spatially inhomogeneous textures. This can be done, for instance, using as a control signal, an auxiliary image, whose local parameters determine parameters of local spectra of textures to be generated.

This idea is implemented in the exercise in the following way. Two types of spectral masks for the LF model are pre-defined: circular masks of different radii and elliptical masks of different orientations. For circular spectral masks, mask radius is the controlled parameter. For elliptical spectral masks, their angular orientation is the controlled parameter. In the process of generating texture images, those parameters are, pixel by pixel, determined by gray levels of pixels of the reference image chosen by the user. Figure 6.11 illustrates the work of the model.

6.4 PWN&LF and LF&PWN models

PWN&LF and LF&PWN models are examples of two-stage (cascade) models with point-wise nonlinearity and linear filter applied to input seed images one after another in different order.

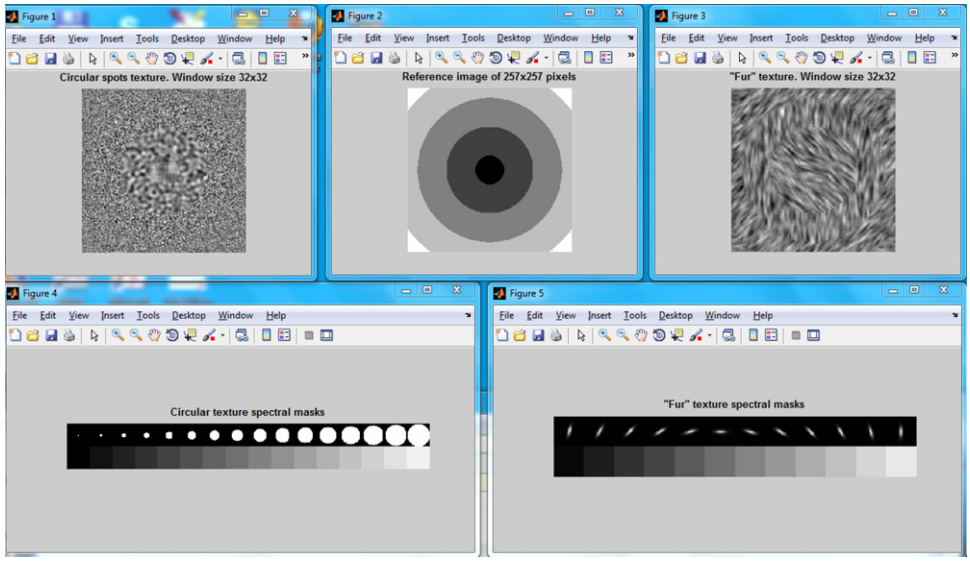

Figure 6.11. LF models: spatially inhomogeneous textures (upper left and right panels) with circular (bottom left) and elliptic (bottom right) local spectra controlled by the reference image (upper middle panel).

A version of the PWN&LF model implemented here imitates 'randomly' tossing an image of a one cent coin onto a plane. To this goal, the model generates in the first stage an array of pseudo-random binary numbers with a certain, set by the user, probability of 'ones'. Then this array is convolved with an image of a cent coin, which plays here the role of the impulse response of the linear filter. As a result, images of cent coins are placed in the positions of ones of the binary array at the linear filter input. The program runs this procedure several times and displays obtained realizations of images as video frames. The number of runs is a user-defined parameter as well. Figure 6.12 shows one frame of such a run.

In the LF&WPN-model, linear filters and point-wise nonlinearities act in the inverse order with respect to the previous case. First, arrays of correlated pseudo-random numbers are produced from a seed array of uncorrelated pseudo-random numbers by a linear filter with a preset frequency response, and then the result is subjected, pixel by pixel, to a certain user-defined nonlinear transformation. The types of filter masks used here are the same as in the above described exercise with the LF model. As for the nonlinearity, two options are offered: threshold non-linearity and sinusoidal one. The threshold nonlinearity replaces positive input signal values by 'ones', and negative values by 'zeroes'. The sinusoidal nonlinearity replaces input signal values by a cosine of two π times signal value normalized to the unity. Figures 6.13 and 6.14 illustrate output displays.

6.5 Evolutionary models

As was mentioned, evolutionary models are iterative models, i.e. models with feedback. They work with the initial 'seed' image in an iterative way, applying, at

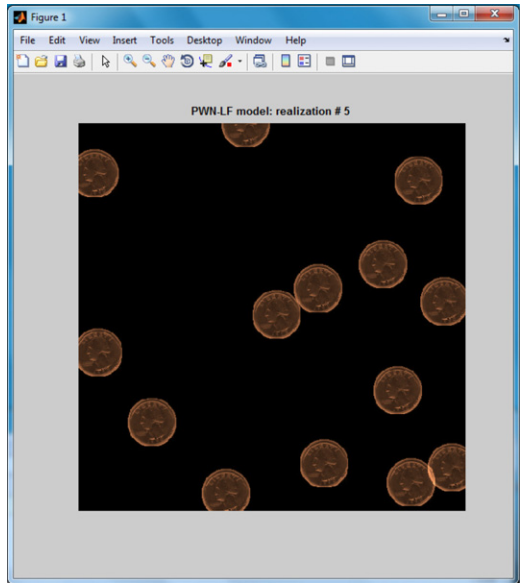

Figure 6.12. PWN&LF model: images of cent coins placed 'randomly' over an empty frame (shown using the MATLAB® color map 'copper').

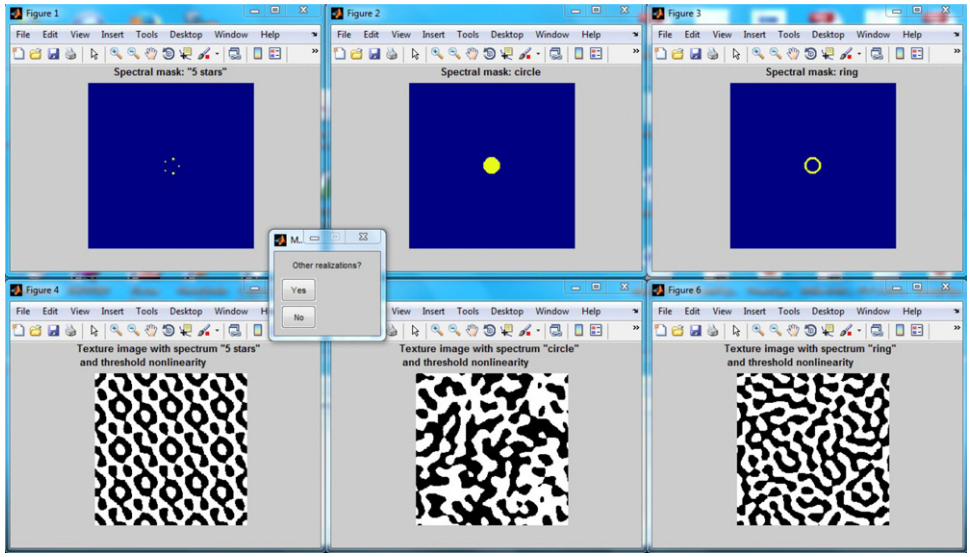

Figure 6.13. LF&PWN model, the threshold nonlinearity. Yellow color in images of spectral masks corresponds to ones and blue color corresponds to zeroes in the frequency responses.

each iteration step, a certain image transformation specific for the model to the result obtained at the previous step. A critically essential feature of evolutionary models is the presence of irreversible non-linearity in the feedback loop. This enables generating nontrivial patterns in the process of iteration, i.e. of evolution of the

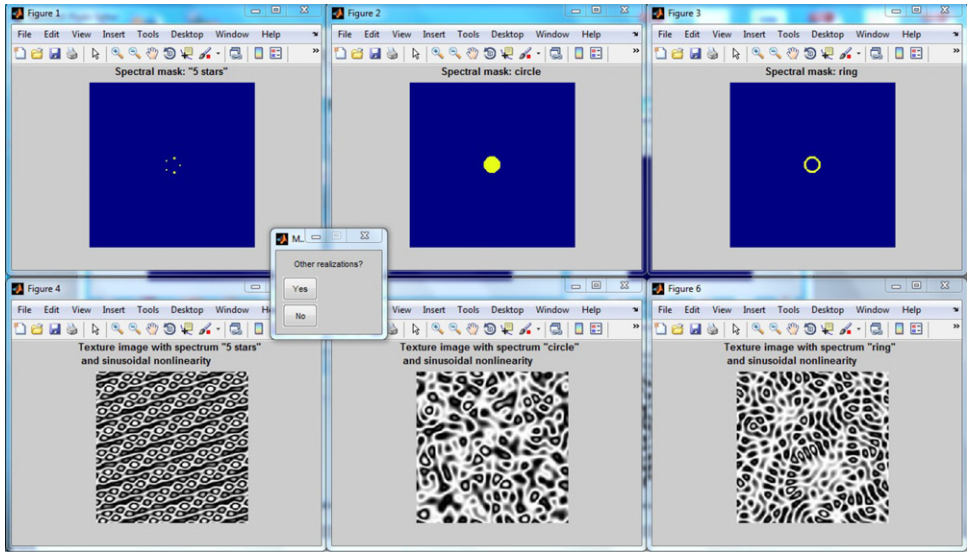

Figure 6.14. LF&PWN model, the sinusoidal nonlinearity. Yellow color in images of spectral masks corresponds to ones and blue color corresponds to zeroes in the filter frequency responses.

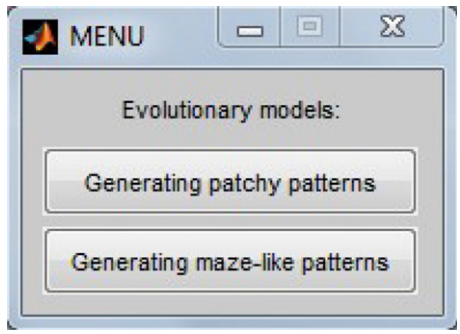

Figure 6.15. Evolutionary models: start menu.

patterns. Two specific models are implemented here: a model that generates patchy patterns and a model that generates maze-like patterns (figure 6.15).

6.5.1 Generating patchy patterns

In this model, images are iteratively subjected to filtering in a running (row-wise and column-wise) window that, at each position of the window, replaces the gray level of the window central pixel by the gray level of the mode of the histogram over the window, i.e. by the gray level, which is the most frequent in the window. This filter belongs to a family of rank filters that will be studied later in chapters 8 and 9. Being applied iteratively, it tends to produce piece-wise constant patchy patterns.

For experimentation with this model, the user is first prompted to choose, as a seed image, either an array of pseudo-random uncorrelated numbers or any natural

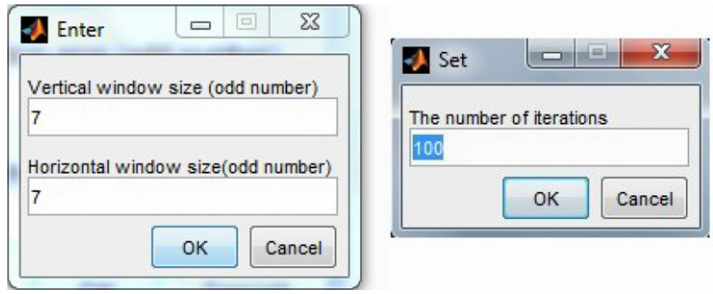

Figure 6.16. Evolutionary models: generation patchy patterns parameter menu.

image from the image database and after that to set the filter window horizontal and vertical dimensions and the number of iterations (figure 6.16).

The program displays the input seed image and, at each iteration, the output image and plots of the number of image pixels modified at this iteration and of the number of histogram bins in the output image versus the number of iteration. The first plot provides an indication of the convergence of the output image to a model 'fixed point', i.e. to a stable image, which, once it has appeared, is not modified anymore in the process of iterations. In this case the number of modified pixels becomes zero. The second plot shows the number of image histogram non-zero bins in the iteration process. Decrease of this number evidences the convergence of the seed image to a patchy piece-wise constant image.

After the chosen number of iterations is executed, the user has an option either to further continue iterations or to stop iterations and display the pattern of edges of the produced patchy pattern. The edges are detected by computing pixel wise differences between output image maxima and minima in a running window of 3×3 pixels. An example of the final display is shown in figure 6.17.

6.5.2 Generating maze-like patterns

The exercise in this section implements a stochastic modification of the mathematical game, the 'Game of life', invented by a British mathematician John Conway. In the standard model 'Game of life', a square array of 'cells', which may be in one of two states, one ('live') and zero ('empty') and forms a binary pattern with pixels representing the cells, evolves from a certain seed pattern according to the three simple rules:

1. If a 'live' cell has in its closest 3×3 pixel spatial neighborhood less than two or more than three 'live' cells, it will 'die' at the next evolution step, i.e. one will be replaced by zero in the corresponding pixel.
2. If an 'empty' cell has in its closest 3×3 pixel neighborhood exactly three 'live' neighbors, this cell will give 'birth' at the next evolution step, i.e. zero will be replaced by one in the corresponding pixel.
3. Otherwise nothing happens.

The 'births' and 'deaths' events in the standard model happen at each evolution step synchronously in all cells of the array, where they must happen according to the

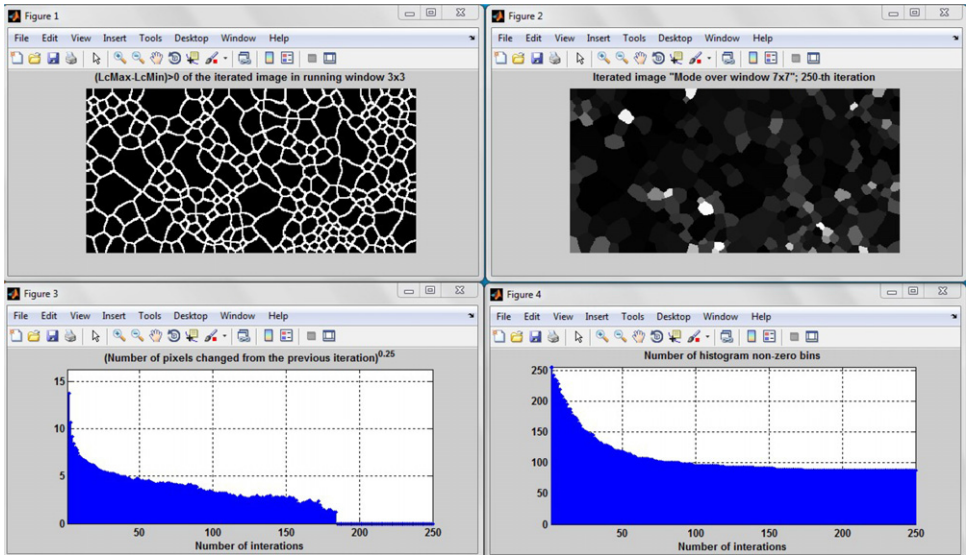

Figure 6.17. Evolutionary models: iterative local histogram modes over a running window. The graph of the number of pixels changed from the previous iteration shows that the model reached its fixed point at about the 170th iteration.

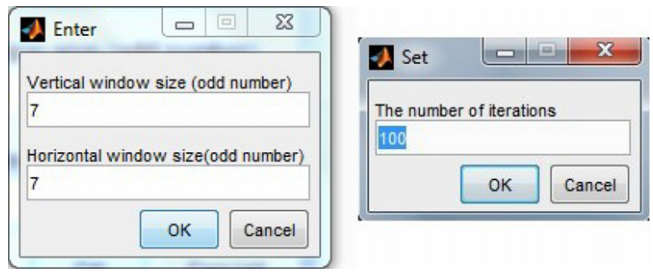

Figure 6.18. Evolutionary models, generating maze-like patterns: parameter menus.

above rules. In the stochastic modification of the model that is offered for experimentation here, the 'birth' events occur in all cells of the array, where they must occur according to the above rule, whereas 'deaths' occur only in a subset of the relevant cells. Pixels of this subset are selected, using a pseudo-random number generator, with a certain probability, which we call the 'probability of death' P_{death}. This is a model parameter that the user has to set along with the initial probability of 'live' cells in the binary pseudo-random seed pattern that the program will generate at its start (figure 6.18).

For the standard, non-stochastic, model $P_{\text{death}} = 1$. The standard 'Game of life' model produces, in the course of evolution from an arbitrary seed pattern, three types of patterns: (i) stable formations that, once appeared, remain unchanged unless they collide with neighbor formations, which might happen in the course of evolution; (ii) periodical formations ('oscillators'), which repeat themselves after a certain number of evolution steps; obviously, the above-mentioned stable patterns

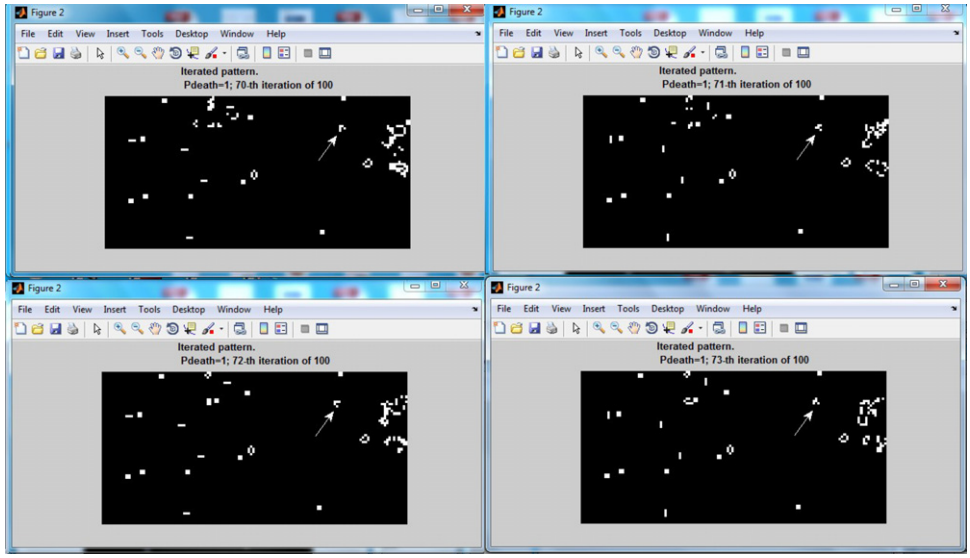

Figure 6.19. Game of Life standard model: four consecutive frames of the evolutional pattern, in which stable, periodical and moving patterns are present. The latter are marked by the arrow.

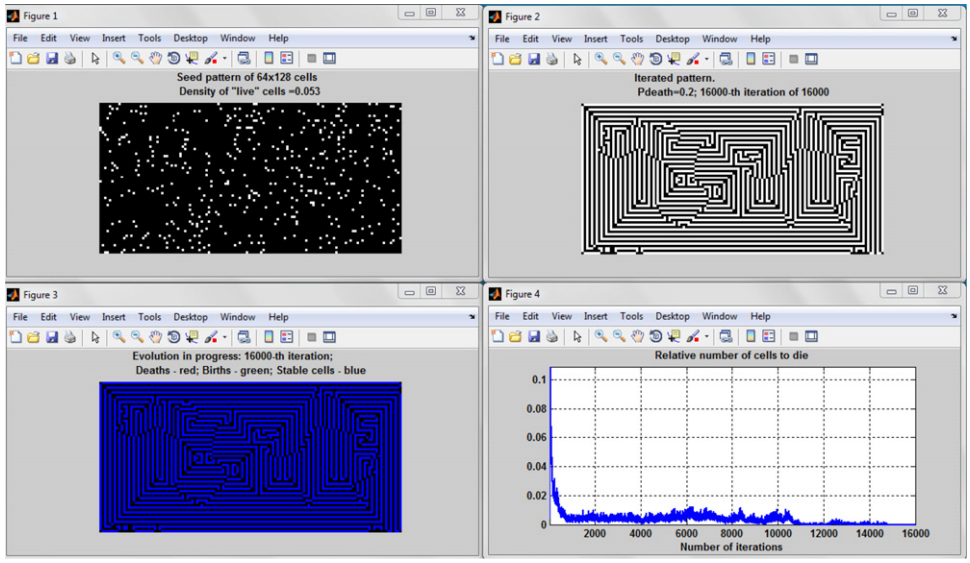

Figure 6.20. Evolutionary models: stochastic modification of the 'Game of Life'. The bottom image, in which stable cells are shown in blue, and plot evidence that the model reached a fixed point after about 15 000 iterations.

can be regarded as a special case of periodical formations with a period of one step; (iii) self-replicating 'moving' formations, or 'gilders', which move across the lattice and replicate themselves in a shifted position after a certain number of steps; this can be regarded as a general 'space–time' periodicity.

The program that implements the model displays, at each iteration, i.e. evolution step, the initial seed pattern, the output pattern and additionally a color coded output pattern, in which cells that are to 'die' on the next evolutionary step are shown in red, cells that will give 'birth' are shown in green, and stable cells are shown in blue. Displayed is also a plot of the relative number of cells that are to die. The plot serves as an indicator of convergence of the pattern to a stable fixed point. An example of four consecutive frames of the evolutional pattern observed for $P_{\text{death}} = 1$ is shown figure 6.19.

For P_{death} larger than about 0.3, the model produces either isolated stable, or oscillated or moving formations such as those shown in figure 6.19, or chaotic patterns that do not stabilize however large the number of evolutionary steps. For values of P_{death} lower than ~0.3, patterns of chaotic formations that emerge at first evolution steps tend, in the course of evolution, to stabilize and to gradually produce stable homogeneous patterns with a certain degree of order, which look like maze patterns of alternative straight vertical and horizontal lines that chaotically switch their direction and/or positioning. These patterns are the model 'fixed points'. Figure 6.20 illustrates an outcome of one of the runs of the model with $P_{\text{death}} = 0.2$.

Questions for self-testing

1. What is the principle of the algorithmic approach to models for generating patterns?
2. Which statistical characteristics of images are controlled by PWN and LF models?
3. How can PWN and LF models be used for imitating different types of image noises?
4. Do 'fractal' textures with $1/f^P$-type spectra and spatially inhomogeneous textures generated by LF models resemble any natural textures?
5. Does the appearance of textures generated with the use of LF models demonstrate association with the shape of filter masks of the models?
6. What kinds of natural texture can be imitated by the PWN and LF models and their combinations and what cannot?
7. What irreversible nonlinearity is present in the model that generates patchy patterns?
8. Define, in terms of the algorithmic approach, a model that produces 'random net' patterns such as patterns of edges of patchy piece-wise constant images.
9. Do maze-like patterns generated by the evolutionary 'Game of life'-based model resemble some natural patterns?
10. What irreversible nonlinearity is involved in the model that produces maze-like patterns?
11. Suggest some other multi-level and multi-branch algorithmic models as an extension of those implemented in the exercises.