

Steady-state image processing

Mario Köppen^{a,*}, Javier Ruiz-del-Solar^{b,1}

^a Department of Pattern Recognition, Fraunhofer IPK Berlin, Pascalstr. 8-9, 10587 Berlin, Germany

^b Department of Electrical Engineering, Universidad de Chile, Casilla 412-3, 6513027 Santiago, Chile

Accepted 29 March 2001

Abstract

This paper presents a new approach to the application mode of image processing operators, the so-called steady-state image processing. The approach reminds a steady-state genetic processing of images by considering each pixel of the image as an individual. So, some pixels are selected, processed and copied back into the image. This differs from the standard approach, where all image pixels are processed at once. The proposed approach offers many choices for variation, and allows for the assignment of dynamic measures to images. This will serve new families of soft computing methods as, e.g. immune-based algorithms, which need images as non-static objects in order to fulfill reasonable tasks. This paper also introduces some basic steady-state operators and exemplifies the analysis of an image by means of a small example. Also, it is shown how steady-state image processing can be applied in the context of texture segmentation. Steady-state image processing can be considered a way of processing images, which is deeply inspired by genetic algorithms. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Soft image processing; Linear pixel shuffling; Autopoiesis; Steady-state genetic algorithms

1. Introduction

Images as processing objects for soft computing algorithms often faces the limitation of being static objects. In general, an image function I that maps the domain of the image (usually a rectangular subset of $N \times N$) into a set of so-called pixel values (usually the set $G = \{0, \dots, 255\}$ for grayvalue images, or G^3 for color images), is all what is needed for a complete specification of an image. Hence, soft computing algorithms may treat an image as every other kind of functional mapping.

An example for such a processing is given in [1], which presents a genetic programming-based framework for the generation of texture filters. Another example is given in [2], where the internal computations of a multilayer perceptron are considered a convolution operator. Then, the training of the neural network gives an adapted convolution filter as well. There are many more examples, but a complete reference would go beyond the scope of this paper.

To regard for the upcoming of new algorithm families within the field of soft computing, as immune based algorithms [3–6] or autopoietic processing [7,8], it has to be noted that such approaches refer to non-static objects by their very nature. As an example, the animal immune system is basically monitoring processes and not objects. Also, a genetic algorithm emulates the fate of a population of individuals, each of which is struggling against habitual conditions by means of increasing its fitness.

* Corresponding author. Tel.: +49-30-39006200;
fax: +49-30-3917517; URL: <http://vision.fhg.de/ipk/koeppen>.
E-mail addresses: mario.koeppen@ipk.fhg.de (M. Köppen),
jruizd@cec.uchile.cl (J. Ruiz-del-Solar).

¹ URL: <http://tamarugo.cec.u-chile.cl/aabdie/jruizd>.

This paper gives a proposal for a manner of processing images, which may obtain an underlying dynamic nature of the image function. This approach will be referred to as steady-state image processing, thus resembling the basic mode of population treatment used for steady-state genetic algorithms: instead of replacing all individuals of a population at once by the best children of the next generation, only a part of the population is processed and replaced. A similar modification will be made in the mode of applying image neighborhood operators.

Since the early days of pattern recognition, image neighborhood operators (short: image operators) played an important role in the conception and design of image processing algorithms [9]. For each pixel of a grayvalue or color image, an image operator specifies two things: it assigns a neighborhood to each pixel (in most cases the four or eight nearest neighbors in the image grid), and it provides a computation with the gray- or color values of all pixels within the neighborhood of the pixel as input and a new gray- or color value as output. Applying an image operator equals an image-to-image operation. The image operator is applied to each pixel of the image “at once”, i.e. the new value at position (x, y) of the result image is computed from the neighbors of pixel (x, y) in the input image. Examples for image operators are the Sobel operator, the Laplace operator, convolution operations or the grayscale dilation and erosion operation, but there are more. As a common rule, image operators are so applied that the result at some pixel position p_1 is not influenced by the result for any other pixel position p_2 .

Steady-state image processing, the formal definition of which will be given in Section 2, modifies this approach by processing only a subset of the image at once, thus allowing for a possible influence of the processing at pixel position p_1 on the result at pixel position p_2 later on. Instead of a single image-to-image operation, now one gets a processing sequence I_n of images (best represented as a movie file). The basic advantage is that the concept of an *image* dynamic can be introduced. This is done by tracking the evolution of a feature, which is computed from the I_n . So, dynamical measures can be computed, and dynamic states (equilibrium, breakdown, even chaos) assigned. Frankly spoken: how well the image resists the “attacks” of the repeated application of an steady-state image operator may serve as qualifying feature for this

image itself. Moreover, it may reflect the organization of an image, and not just its structure, since it modifies the relations between image parts.

However, the study of steady-state image operators is just at its beginnings, and the underlying mathematics may become quite complicated. So far, we are going to present some examples for steady-state image operators and demonstrate, how they treat certain image classes, and provide some choices for dynamic measures derived from images this way.

Note that the presented approach can be related to cellular automata, and insights from this field can be used to get a better understanding of the steady-state processing. The important difference is that the sequence of states is a sequence of images as well, hence the state changing operators may be defined in terms of image features.

Also, steady-state image operators generalize the well-known relaxation operators. During relaxation, a pixel of an image is modified. Then, according to an energy measure of the whole image it is decided whether the modified pixel is preserved (if it improves the energy value) or if the modification is rejected.

In the following, in Section 2 steady-state operators are formally defined, then, in Section 3, some example operators are introduced and discussed. Section 4 gives a short example of the analysis of an image, using the concept of dynamic measures. Then, Section 5 gives a short insight into a real-world application consisting in the segmentation of textures using the image-processing paradigm here proposed. Finally, Section 6 suggests some alternative ways for specifying the operators involved in the steady-state processing and gives an outlook of this work.

2. Definition

Be I_n a sequence of images, considered as a series of states of one image I_0 , which is equal to the input image. Given I_n , the image I_{n+1} is derived by the specification of the steady-state image operator by means of a tuple of four operators $(S(I_n), N(I_n), O(I_n), R(I_n))$. At first, image I_{n+1} is made from a copy of image I_n . The operator $S(I_n)$ is the specification for selecting a set U of pixels within I_n , and it may depend on the current state of the image. Then, $N(I_n)$ assigns a neighborhood to each pixel in U , and the computation

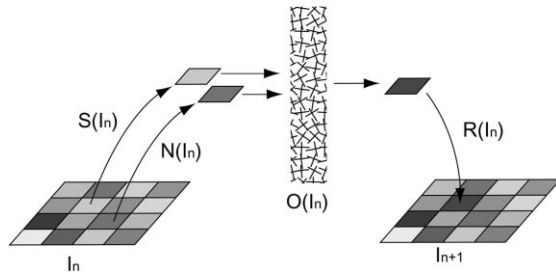


Fig. 1. General procedure of the steady-state image processing.

given by $O(I_n)$ is applied to all gray- or color values of those neighborhoods independently. This gives a set of $|U|$ new gray- or color values. Specification $R(I_n)$ indicates, where to put these new gray- or color values in the image I_{n+1} . All other pixels in I_n and I_{n+1} are equal. For an illustration of the general procedure, see Fig. 1.

3. First collection of steady-state image operators

With the following definitions, basic approaches to the steady-state image processing are supplied. In the following, none of the operators S , N , O or R really depends on the image I_n , i.e. they are selected independently from the image. The operator S simply selects one pixel location in the image, with uniform distribution of the x - and y -coordinates (this gives a non-uniform distribution of the (x, y) locations). The operator N selects a random neighbor p_2 of the se-

lected pixel $p_1 = S(I_n)$ within the four-neighborhood of p_1 . The only varying operation is O , and the formal definitions of $r = O(p_1, p_2)$ are given within the following subsections. Then, the R -operation is to put the value r onto the position of p_1 in order to obtain image I_{n+1} from I_n .

For demonstration purposes, three test images have been used. Image 1 is the well-known Lena image, which is an example of a scene image. Image 2 is a patchwork of two parts of an EC bankcheck with parts of handwriting on it. This is an example for a texture image. Image 3 is the Kirlian photography of a fingertip, providing an example for a nearly binary image.

In Figs. 2, 3 and 5, the effects of the proposed operators after a greater number of iterations can be seen. Also, there is the SSIP website, where one can find Quicktime movies of the steady-state mode of image operation. The URL is <http://vision.fhg.de/ipk/demos/ssip>.

With the exception of the Kirlian image, to which the application of the Xor and Xor3 operator was not reasonable, all operators were applied to all three images.

As a common rule, all proposed steady-state image operators have a more destructive influence on the image content. After a sufficient large number of iterations, the former image content will be more or less destroyed. Image analysis has to be based on the “behavior” of the image in the beginning of the processing. As a common rule, for an image of about 100×100 pixels, this will be up to about 500,000

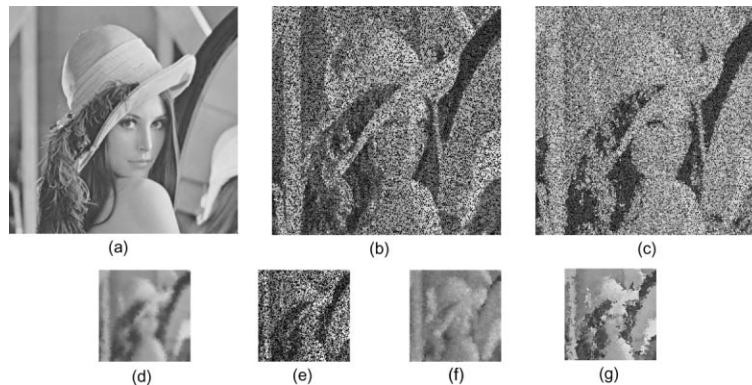


Fig. 2. Steady-state processing the Lena image: (a) original; (b) Xor; (c) Xor3; (d) average; (e) scaled gradient; (f) scaled maximum; (g) transduction.

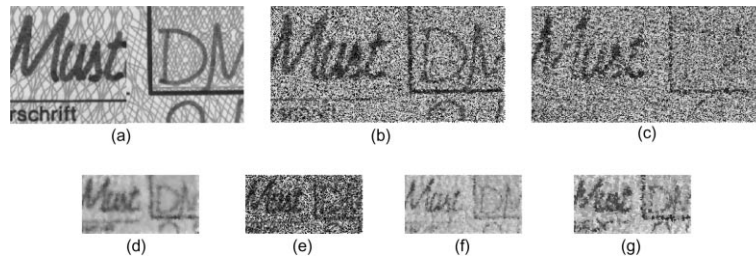


Fig. 3. Steady-state processing the EC-check image: (a) original; (b) Xor; (c) Xor3; (d) average; (e) scaled gradient; (f) scaled maximum; (g) transduction.

iterations, and this value increases with image size n by order $O(n^2)$.

3.1. Xor operation

The steady-state Xor operator was already introduced in [7], and it was proven to be the only auto-projective operator. The steady-state Xor operator is defined with the following formula:

$$r = I_n(p_1) \oplus I_n(p_2), \quad (1)$$

where ' \oplus ' stands for the bitwise Xoring of the two grayvalues.

In [7], the importance of this operator for obtaining a computational model for autopoietic processing of images was already pointed out. The argument was based on the consideration of reproductive properties of a possible steady-state operation mode for applying image operators. The Xor operator has the following property:

$$(a \oplus b) \oplus b = a, \quad (2)$$

which allows for possible stable structures in images. The results given in Figs. 2(b) and 3(b) and on the SSIP website demonstrate a more "salt-and-pepper" processing of images, but this impression is not correct. While there is a positive probability of reproducing pixel grayvalues, the location where this reproduction occurs might shift slightly.

This fact is illustrated in Fig. 4. The effect of steady-state processing of the face image (a) for 50,000 iterations is shown in subimage (b). After applying a dilation to the result image (b), which gives subimage (c), the image (a) seems to be nearly recovered. The only differences are the borders around the Phong-like structures in the face image (a).

3.2. The Xor3 operator

While the only operations with an orbit of size 2 (as the Xor operation with the property $a \oplus a = e$, with e being the identity operation) for any set of possible grayvalues is the bitwise Xor operation, there

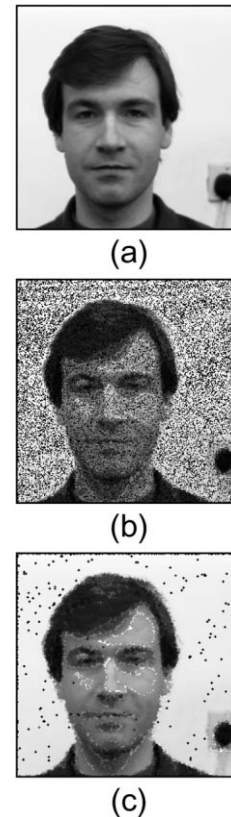


Fig. 4. Application of dilation (c) to the steady-state Xor processed image (b) nearly recovers the starting image (a).

Table 1

The operation table for a group operation \oplus_3 , for which each element has an orbit of size 3

	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

can be designed similar operations with other prime numbers as orbit sizes. From a basic theorem of group theory (for a proof consider [7]), then the order of the corresponding group established from the set of grayvalues as elements and the operation ‘ \oplus ’ as group operation, the order of the group must be a power of the size of the orbit of an element, as long as this size is prime. Hence, all groups with “reproductive” properties, i.e. with the property $a \oplus a \cdots \oplus a = e$ can be reduced to a bitwise operation mode of the basic operation for a group with exactly p elements (with p being the prime number of the size of the orbit of an element). For the case $p = 3$ this gives a unique operation on a group of three elements $G = \{0, 1, 2\}$ with the property $a \oplus a \oplus a = e$ for each $a \in G$.

It can be easily shown that such a group has the operation table for \oplus_3 as given in Table 1.

From this, a new steady-state operator Xor3 can be easily designed. Since the order of the group has to be a power of 3, the 256 grayvalues of a standard grayvalue representation of an image were rescaled to the grayvalue range $0, \dots, 242 = 3^5 - 1$. Then, the grayvalues at positions p_1 and p_2 are represented to the number base 3 and the operation \oplus_3 , as given in Table 1, is applied digitwise:

$$r = I_n(p_1) \oplus_3 I_n(p_2). \tag{3}$$

Examples for the effect of this operation, which are similar to the Xor operation, are given in Figs. 2(c) and 3(c) and on the SSIP website.

3.3. Average operator

The average operator uses the O -operator:

$$r = \frac{I_n(p_1) + I_n(p_2)}{2}, \tag{4}$$

i.e. the new value replacing the grayvalue at the randomly selected position p_1 is the average of the grayvalues at positions p_1 and p_2 .

The average operator has an equivalent standard image operation, which computes the average grayvalue of each image pixel and its neighborhood at once. Thus, the effect of the steady-state average operator is not very different from the repetitive application of a standard average operator on images.

The effect of the steady-state average operator on our demonstration suite can be seen in Figs. 2(d), 3(d) and 5(b) and the corresponding movie files on the SSIP website.

3.4. Transduction operator

The steady-state transduction operator appears to be the most simple one: a randomly selected pixel is replaced by one of its neighbors. More formally:

$$r = I_n(p_2). \tag{5}$$

This operation has the image shift as standard (i.e. non-steady-state) image processing counterpart, but is a completely different modification of the image. From the examples in Figs. 2(g), 3(g) and 5(e) and the SSIP website it can be seen that the kind of processing resembles a more fractal “melting” of the image content.

An interesting variation of this operation is the conditional transduction (more a relaxation operation) with the O -operator:

$$r = \begin{cases} I_n(p_1) & \text{for } |I_n(p_1) - I_n(p_2)| \geq \sigma \\ I_n(p_2) & \text{for } |I_n(p_1) - I_n(p_2)| < \sigma, \end{cases} \tag{6}$$

i.e. the value at p_2 is only copied over the value at p_1 if the difference in grayvalues is lower than a threshold σ . Fig. 6 shows the effect of this operation after performing 100,000 steps for various values of σ . For smaller σ -values, the operation tends to be a segmentation of the image, since the processing will not cross over image boundaries with a higher gradient.

3.5. Scaled maximum operator

The scaled maximum operator in steady-state operation mode is given with the equation:

$$r = \alpha \max[I_n(p_1), I_n(p_2)]. \tag{7}$$

The important aspect here is the use of a parameter α with values from the range $[0, 1]$. While for

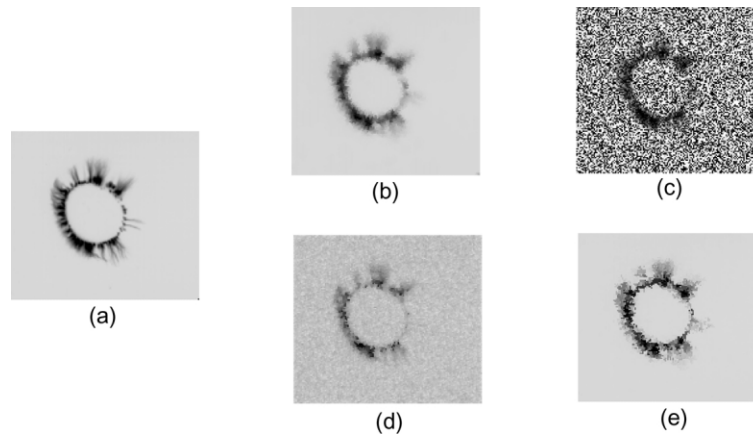


Fig. 5. Steady-state processing the Kirlian image: (a) original; (b) average; (c) scaled gradient; (d) scaled maximum; (e) transduction.

$\alpha = 1$ the result would be just a rapid distribution of the brighter pixels in the image over the whole image area, for smaller values of α ($0.9 < \alpha < 1.0$) this process may become elongated. For even more smaller values of α , the image will become dark at the end. An example analysis of the scaled maximum operator, based on several values of α , will be presented in Section 4.

The effect of steady-state scaled maximum processing of images can be seen in the Figs. 2(f), 3(f) and 5(d) and the movie files on the SSIP website.

Using ranking, minimum, ordered weighted averaging or T- and S-norms instead of the maximum will provide further steady-state operators as well.

3.6. Scaled gradient operator

This operator specifies the O -operator by the formula:

$$r = \alpha \|I_n(p_1) - I_n(p_2)\| \quad (8)$$

with α being a positive parameter (reasonable values can be taken from the interval (1.0, 1.2)).

The steady-state scaled gradient operator rapidly destroys the image structure, as can be seen from the examples in the Figs. 2(e), 3(e) and 5(c) and the SSIP website. However, the images will not converge to a stable final structure, but rather to a kind of salt-and-pepper noise image. This can be understood from considering the limiting cases: when the two

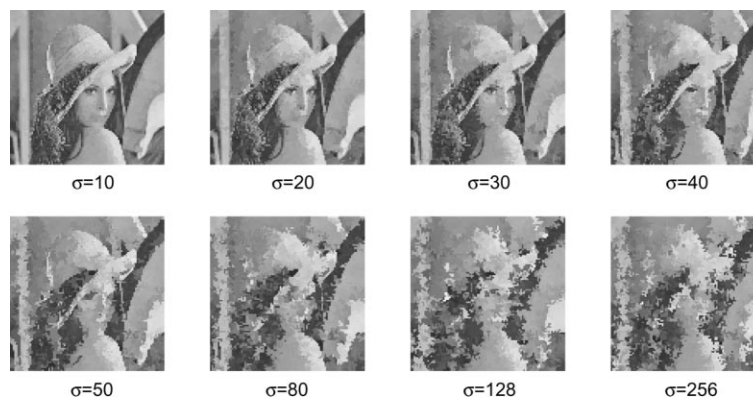


Fig. 6. Effect of conditional transduction after 100,000 iterations for various values of σ .

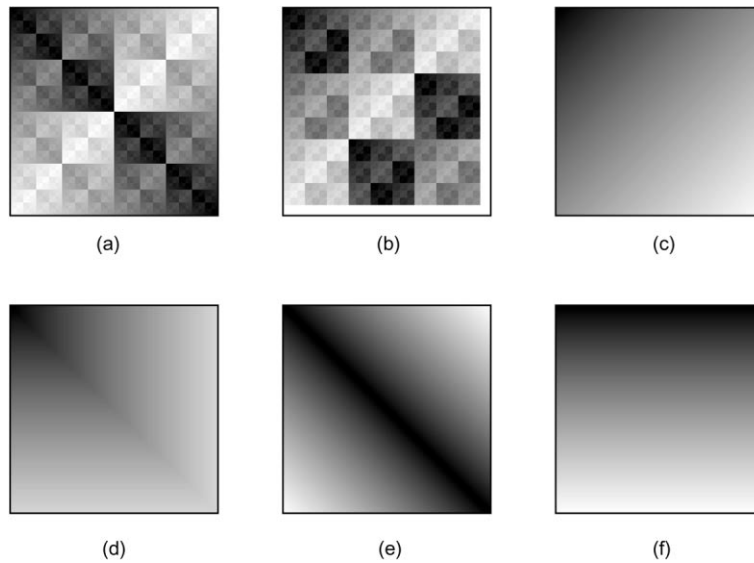


Fig. 7. Lookup matrixes for the presented steady-state operators: (a) Xor; (b) Xor3; (c) average; (d) scaled maximum ($\alpha = 0$); (e) scaled gradient ($\alpha = 1.0$); (f) transduction. Note the smaller size for the lookup matrix of the Xor3 operator.

neighboring pixels selected both are bright, the value of r will be low and p_1 will be replaced by a dark pixel, thus putting a bright and a dark pixel in direct adjacency. When this pair is selected again, the gradient is high, and the former dark pixel p_1 now is replaced by a bright pixel, re-establishing the initial situation. Thus, there is no stable configuration for the scaled gradient operator, and the image structures are “destroyed” starting off from its more homogeneous parts (where gradient values are small).

3.7. General characterization

The given example operators can be unified in a general approach, which is based on a lookup matrix L . Thereby, L is a grayvalue image of dimension $g_{\max} \times g_{\max}$, with g_{\max} being the maximum grayvalue of the images I_n . Then, the O -operator is given by:

$$r = L(I_n(p_1), I_n(p_2)). \quad (9)$$

Fig. 7 gives the corresponding lookup matrixes for the steady-state operations presented in the foregoing subsections.

As it can be easily seen, this gives a simple representation of steady-state operators, for which the O -operator does not depend on the step n .

Fig. 8 gives an example for a steady-state operator defined by an arbitrary lookup matrix. An interesting aspect here is the obvious stability of the processed image after about 100,000 steps.

4. Example image analysis

As already mentioned, one important aspect of the steady-state approach to image processing is the possible assignment of dynamical measures to images. The most prominent example for such a measure is the image volume, i.e. the sum of the grayvalues running over all possible pixel positions.

The image volume will be acquired for say all 1000 iterations. In Fig. 9, three plots of the image volume measure are given for the steady-state scaled maximum operation (see Section 3.5) applied to the Lena test image. The plots show the image volume against the number of iterations in units of 1000 steps. Three different values for α were used. For convenience, the processed image after 100,000 iterations is supplied as well. As can be seen from the plot, for $\alpha = 0.95$ the image volume remains nearly constant for a larger number of iterations, while it rapidly becomes small for $\alpha = 0.8$ and becomes high for $\alpha = 1.0$. Thus, 0.95 can be considered a characteristic

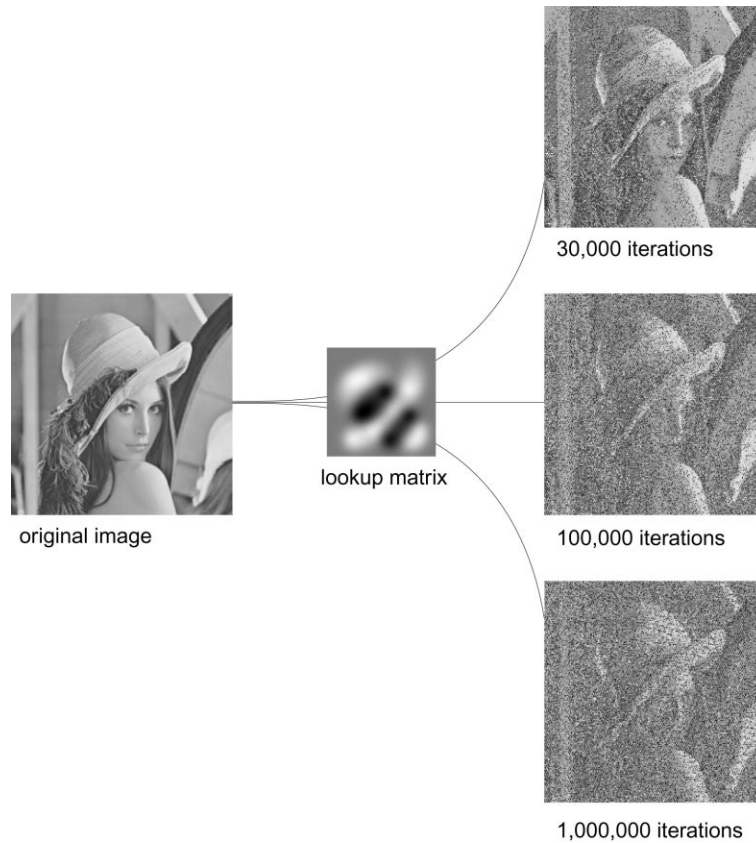


Fig. 8. Steady-state operator defined by an arbitrary lookup matrix.

value (or feature) of the Lena image (in given resolution).

It is obvious that such kind of analysis will work for every image. For small α , the image will become completely dark after a sufficient large number of iterations, and for α near 1, it will contain only white pixels. Hence, there must be a range for values of α , where at least a temporary stability may occur. Such “behavior” of an image can be used for characterizing the image as well.

This may serve to illustrate the basic idea of an image analysis based on steady-state image processing.

5. Real-world application: texture segmentation using SSIP-operators

As a first real-world application of the here-described image-processing paradigm, we are going

to present the segmentation of natural textures using SSIP-operators. The operator selected to perform this task was the scaled minimum, defined as:

$$r = \alpha \min[I_n(p_1), I_n(p_2)] \quad (10)$$

This operator is applied to the textured image being segmented, and after an interaction time the segmented image emerged. The result of the application of this operator to two natural textures from the Brodatz Album [10] is shown in Fig. 10. The both selected textures are very difficult to segment with standard methods (co-occurrence matrix, Gabor-based, wavelets, etc.). As it can be seen in Fig. 10(c) and (d) the segmented results are satisfactory and even suitable to be improved using nonlinear post-processing. At this time we are working in this direction.

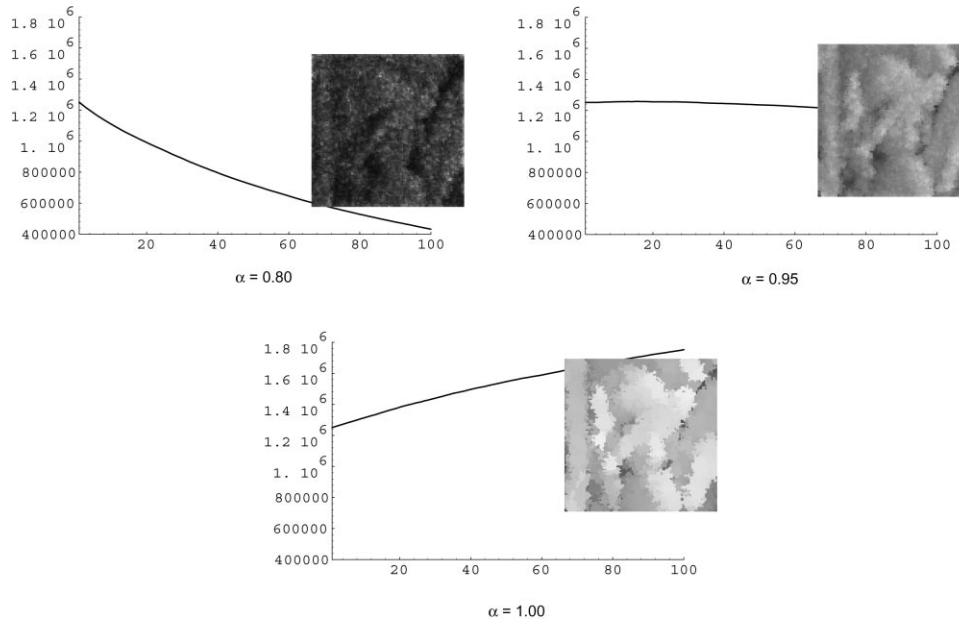


Fig. 9. Plot of image volumes for the steady-state scaled maximum operation for the first 100,000 iterations.

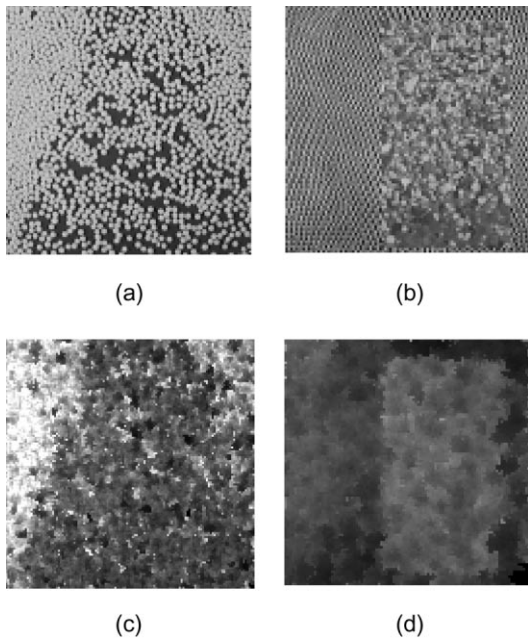


Fig. 10. Segmentation of two Brodatz natural textures ((a) and (b)). The segmented textures after 10,000 iterations are shown in ((c) and (d)). The α parameters being utilized are 1.15 and 1.05, respectively, for images (c) and (d).

6. Further suggestions and outlook

There are many choices for the operators included in the specification of a steady-state operator, and many ways to make them dependent on the actual image. For $S(I_n)$, the choice is strongly related to the decision for a random distribution of the selected pixels. For obtaining a uniform distribution of the pixels (not the coordinates), a procedure like the linear pixel shuffling (LPS) [11] can be used. However, the LPS has been proven to increase processing speed for image processing operations, since a processing result similar to the image processed in a standard way (all pixels at once) may be obtained with a remarkable smaller number of steps.

So, it is not expected to obtain a qualitatively different image in the steady-state case when LPS is used as S -operator.

Other possible specifications of a steady-state image operator include, but are not restricted to: giving $S(I_n)$ by Markov random fields or by selecting pixels according to a random distribution (which might depend on I_n); giving $N(I_n)$ and $O(I_n)$ by one of the many known image neighborhood operators; giving $R(I_n)$ by selecting the same points as $S(I_n)$, or using distinct Markov random fields.

It should be mentioned that Markov random fields are widely used in the field of texture synthesis. Taken this fact into account, we believe that steady-state image processing has many applications on the synthesis of textures. At the moment we are working in this subject.

Those variations and the issues related to their mathematical modeling and their versatility are the subject of further research.

The basic concept, to provide a manner of image processing inspired by genetic algorithms (and thus inspired by biology) offers a great choice of new ways for applying soft computing technologies in the field of image processing.

Acknowledgements

This research was supported by FONDECYT (Chile) under Project Number 1990595 and by the joint “Program of Scientific Cooperation” of CONICYT (Chile) and DFG (Germany).

References

- [1] M. Köppen, M. Teunis, B. Nickolay, A framework for the evolutionary generation of 2d-lookup based texture filters, in: T. Yamakawa, G. Matsumoto (Eds.), *Methodologies for the Conception, Design and Application of Soft Computing*, Proceedings of the 5th International Conference on Soft Computing and Information/Intelligent Systems (IIZUKA'98), 1998, pp. 965–970.
- [2] H. Watabe, K. Arakawa, Y. Arakawa, Cascaded nonlinear filters and its application to image processing, in: *Proceedings of the 4th Asian Fuzzy Systems Symposium*, Tsukuba, Japan, 2000, pp. 120–125.
- [3] A. Somayaji, S. Hofmeyr, S. Forrest, Principles of a computer immune system, in: *Proceedings of the New Security Paradigms'97*, Great Langdale, Cumbria, UK, 1997, pp. 75–82.
- [4] S. Forrest, S.A. Hofmeyr, A. Somayaji, Computer immunology, *Communications of the ACM* 40 (10) (1997) 88–96.
- [5] H. Bersini, F.J. Varela, The immune recruitment mechanism: a selective evolutionary strategy, in: R.K. Belew, L.B. Booker (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 1991, pp. 520–526.
- [6] T. Shimooka, Y. Kikuchi, K. Shimizu, Study on the dynamics of the idiotypic network for the feature extraction of patterns, in: *Proceedings of the International Workshop on Soft Computing in Industry'99 (IWSCI'99)*, Muroran, Hokkaido, Japan, 1999, pp. 313–318.
- [7] M. Köppen, J. Ruiz-del-Solar, Autopoiesis and image processing I: detection of image structures by using auto-projective operators, in: M. Mohammadian (Ed.), *Computational Intelligence for Modeling, Control and Automation*, Vienna, Austria, 1999, pp. 66–71.
- [8] J. Ruiz-del-Solar, M. Köppen, Autopoiesis and image processing II: autopoietic-agents for texture analysis, in: M. Mohammadian (Ed.), *Computational Intelligence for Modeling, Control and Automation*, Vienna, Austria, 1999, pp. 72–76.
- [9] D. Marr, *Vision*, MIT Press, New York, 1981.
- [10] P. Brodatz, *Textures*, Dover, New York, 1966.
- [11] P.G. Anderson, The unit RBF network: experiments and preliminary results, in: M. Heiss (Ed.), *Proceedings of the International ICSC/IFAC Symposium on Neural Computation (NC'98)*, International Computer Science Conventions (ICSC), Canada/Switzerland, 1998.