# Deriving Rules from Evolutionary Adapted Texture Filters by Neural Networks

Mario Köppen, Achim Zentner, Bertram Nickolay
Fraunhofer-Institut IPK Berlin
Department of Pattern Recognition
Pascalstr. 8-9, D-10587 Berlin, Germany
mario.koeppen@ipk.fhg.de

March 31, 2000

**Abstract**

The work presented in this paper is motivated by the recently proposed 2D-Lookup framework for the evolutionary and data-driven adaptation of texture filters. A class of images, the 2D-Lookup matrices, appears to play an important role for the performance of the adapted texture filters. Two approaches for approximating these 2D-Lookup matrices by neural networks are presented, one based on the Multilayer Backpropagation Neural Network (MBPN), and one based on the Unit RBF network. For the MBPN approach approximating 2D-Lookup matrices very smooth but rather rough, the Unit RBF approach approximates these images much more better, especially at specific details at a lower scale. Also, the Unit RBF approach is faster and more simple to handle, and its outcome serves a texture model based on fuzzy rules.

## 1 Motivation

Recently, a framework for the automated generation of texture filters using evolutionary computation, was presented [**?**]. The framework employs the so-called 2D–Lookup algorithm, which goes as follows.

Given a texture image containing a foreground structure (e.g. a texture fault, handwriting). This input image is assumed to be a grayscale one, with each grayvalue from the set $\{0,1,\ldots,255\}$ (consider figure 1). The task is to design a filter, which separates the foreground structure from its textured background. The task is specified by a second binary image, the so-called goal image, which is manually crafted (e.g. by a photo retouching program like Photoshop, consider figure 2 as an example for such a goal image for the image in figure 1). In the goal image, the shape of the foreground structure is painted black, the texture–for–removal region is painted white. This is all, what is given. There is no special texture model known to the user and there are no further requirements for the goal image.

For specifying the 2D-Lookup algorithm, two image processing operations $op_1$ and $op_2$ are to be named. If $I(x,y)$ is the image funktion of the input image, for which the filter has to be designed, the two operations, if applied to $I$, gives two result images $op_1(x,y)$ and $op_2(x,y)$, both of the same size as $I$.

Figure 1: Example for texture image containg fault.

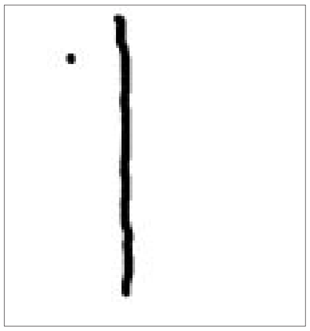


Figure 2: Goal image for the texture fault in figure 1, as given by the user.

Now, as a second specification of the algorithm, a set of (crisp) rules is given. Each rule is parametrized by the image coordinates $x$ and $y$ and gives an entry in the result image of the algorithm according to the grayvalue outcome of the two operations. Such a rule has the general form

$$\begin{array}{c} if\, op_1(x,y)\, is\, g_1\, and\, op_2(x,y)\, is\, g_2 \\ then\, res(x,y)\, is\, lt(g_1,g_2) \in \{0,1\} \end{array}$$

There is a maximum of $256 \times 256$ rules necessary to specify the algorithm. It is convenient to display all rules by means of an image of size $256 \times 256$, the so-called 2D-Lookup matrix, with $lt(g_1,g_2)$ as image function.

Figure 3 gives an example for this procedure, including input image, goal image, result of $op_1$, result of $op_2$, 2D-Lookup matrix and result of the algorithm.

The purpose of the framework presented in [**?**] was to configure the 2D-Lookup algorithm properly just by means of input image and goal image, in order to supply the most simple user interface for texture filter generation. This goal was achieved by using evolutionary algorithms, especially genetic algorithms, genetic programming [**?**], and a variant of the Nessy algorithm capable of multi–objective optimization [**?**]. Each individual of the population of the evolutionary algorithm represents one configuration of the 2D-Lookup algorithm. Therefore, an approach has to be given for the following issues:

- Coding: The individual must specify $op_1$, $op_2$ and the 2D-Lookup matrix $lt$ in order to fully specify the 2D-Lookup algorithm.

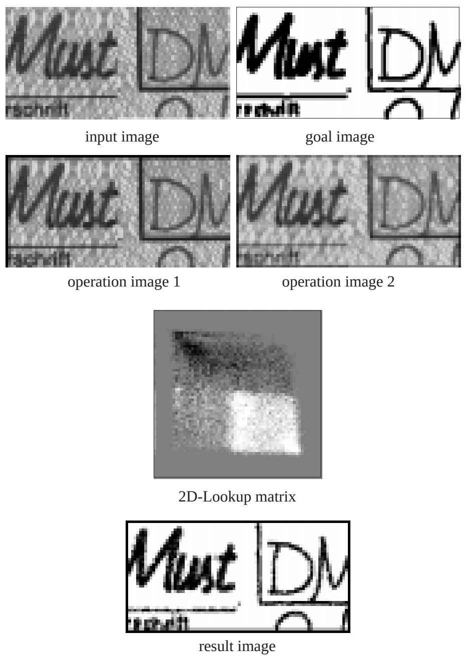


Figure 3: Adapted 2D-Lookup framework applied to bankchecks texture background removal (from [**?**]).

- Fitness: A measure has to be given for comparing the result of the 2D-Lookup algorithm, as configured by an individual, with the goal image.

By using genetic programming (GP), best results were achieved. In GP, each individual is represented by an expression tree, with image operations as node functions. For details of the GP approach see [**?**]. The result of a GP run is a fully specified filter program, which performs the two operations on an arbitrary input image, and the 2D-Lookup matrix.

The issue, which was left open in [**?**] is the generalization ability of the designed filters. While there were designed for one and only one input–goal image pair, it may not be obvious how the filter performs, if the same situation is presented within another image.

The key for checking generalization ability of such a designed filter is given by the generated 2D-Lookup matrices. These matrices can be manipulated for improving the filter's generalization ability. Figure 4 shows some 2D-Lookup matrices, which were the result of applying the framework to various texture fault examples (the texture faults itself are of minor interest for the following). By considering these matrices, the following can be noted:

- Some matrices seems to be separable, i.e. the textured background is represented by a group of compact white regions. The fault appearance (the black dots) circumscribe these regions.

- The gray parts of the matrices represents positions, for which no special rule can be assigned, since this grayvalue constellation was not present within the operated input images.

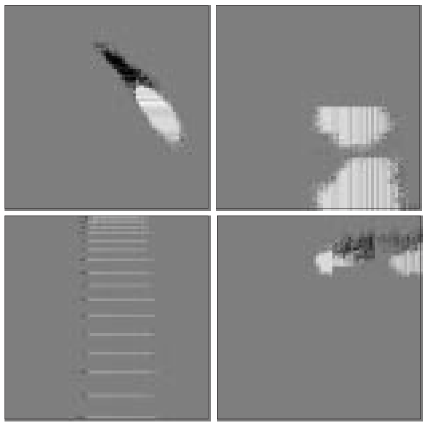


Figure 4: Example for 2D-Lookup matrices generated by the 2D-Lookup framework.

- There are regions, where black and white dots are hoplessly intermingled. The more such regions are found within a matrix, the more the evolutionary adaptation has featured random grayvalue constellations within the two operation images.
- Some matrices are separable by a horizontal or vertical straight line (e.g. lower left example of figure 4). This means, that the 2D-Lookup algorithm can be simplified to thresholding $op_2$ or $op_1$ resp.

To summarize: compactness within the 2D-Lookup matrices is considered as main provision for filter's high generalization ability. If the matrices are manipulated in a manner, which enhances compactness of its black and white regions, the filter will perform better on newly presented images (possibly for the price of a slightly lower performance on the input image, from which the filter was designed).

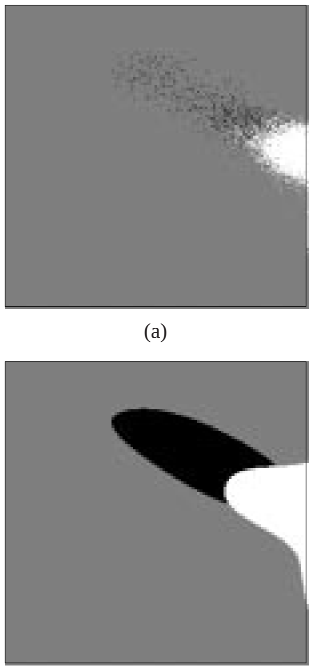
But filter performance is not the only advantage of such a procedure. If it would be possible to provide a description of the compact regions within a 2D-Lookup matrix on a higher level than by its plain pixel sets, this information would allow for deriving texture models from the framework's results.

This leads to a new statement of the problem: to find out about compactness within the class of images, to which 2D-Lookup matrices belong. On a first glance, this seems to be a typical problem for cluster analysis. It is not so for the following reason: the regions may visually form compact clusters, but in some cases this clusters are only given by some pixels, which are spread over the prospective cluster region within the image. In other cases they are given as connected sets of pixels, but with "fuzzy" borders. A procedure for performing such a clustering has to account for both cases. Such a procedure must be able to generalize by itself.

Neural networks would give a suitable procedure, since they attain generalization from data. In the following, two approaches are presented. The first approach uses a multilayer backpropagation network (MBPN), the second approach the unit radial basis function network, as proposed in [**?**].

## 2 The MBPN approach

Training a MBPN by means of a 2D-Lookup matrix goes as follows: From the 2D-Lookup matrix, two images were generated. The image $I_1$ has all positions white (1), which are white in the matrix, all other are set to black. The second image $I_2$ has all positions set to black (0), which are black in the matrix, and the other positions were set to white. Now, $I_1$ and $I_2$ provide training data for training a MBPN on a two–class problem (black and white pixels). The MBPN used has four input neurons, ten hidden neurons and one output neuron. In a training data set, the input is given by a random $(x, y)$ position of the corresponding image, and the output is given by the value of the corresponding image at this position (1 or 0). The four input neurons receive the values $x$, $y$, $x^2$ and $y^2$. The squares are given too for allowing the neural network to use discriminating curves of second order for class separation in its feature space (which is equal to the coordinate plane of the corresponding image).


(a)

(b)

Figure 5: 2D-Lookup matrix (a) and its approximation (b) (simple case).

For each of the two images, 500 positions were randomly selected, and the MBPN was trained over 500 cycles. Then, the trained network was recalled over all image positions, yielding an image, which contains only compact regions due to the separation abilities of a MBPN in general. Therefrom, image $rec_1$ was derived from $I_1$ and image $rec_2$ was derived from image $I_2$. Both recall images are fusioned to a single image $rec$ by the rules given in table 1.

For the second rule, which comprises a conflicting situation where $rec_1$ is white and $rec_2$ is black, white is chosen since in general black dots surround compact white regions. Figure 5 gives the result of such a run. The result is a very good representation of the compact regions of the original 2D-Lookup matrix. Figure 6 shows the result

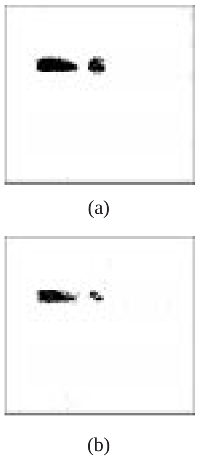


Figure 6: Result image with original 2D-Lookup table (a) and approximated one (b) of figure 5.

Table 1: Rules for fusion of the two recall images at position $(x,y)$.

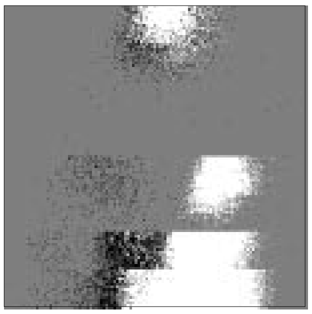
| $rec_1$ | $rec_2$ | $rec$ |
|---|---|---|
| white | white | white |
| white | black | white |
| black | white | gray |
| black | black | black |

images with the original 2D-Lookup matrix and the approximated one.

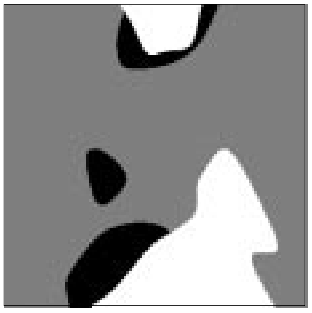
But for rather complicated structured 2D-Lookup matrices, this procedure fails. This is due to a large number of isolated black pixels scattered over a large image region. Preprocessing of $I_2$ with morphological operations helps in this case. The preprocessing steps for obtaining the approximated matrix in figure 7 were: morphological opening with a $3 \times 3$ full mask and morphological closing with a $3 \times 3$ cross-mask. This procedure removes all isolated black pixels and gives “mini-clusters” at fairly scattered regions.

The following can be said about the MBPN approach:

- The MBPN approach is able to approximate 2D-Lookup matrices with useful results.
- The approximation borders are smooth, but the general approximation is rather rough. Some (important) details on a lower scale might get lost in the approximation.
- The general effort of the approach is not very high, but the time for training the two MBPNs is comparable high.
- The weights of the hidden neurons comprise a separation of the coordinate plane of the image. From this, a texture model can be derived, which consists of rules



(a)



(b)

Figure 7: 2D-Lookup matrix (a) and its approximation (b) (complicated case).

with second order inequalities as antecedents. Such a texture model remains a crisp one.

## 3 The Unit RBF Approach

### 3.1 The Unit RBF

Die Unit RBF was proposed by Anderson [**?**]. Its graphical representation is a three layer radial basis function network (RBF), which uses only weights out of the set $\{-1,0,1\}$. The computation of a Unit RBF can be summarized with the formula

$$o(\vec{x}) = \sum_{k=1}^{N} d_k e^{-k||\vec{x}-\vec{c}_k||^2}$$

with $\vec{x} = (x_1, x_2, \ldots, x_n)$ being the input vector, $d_k$ an integer from $\{-1,0,1\}$ and $\vec{c}_k$ an appropriate sequence of Gaussian centers.

The goal of "training" an Unit RBF is to approximate a set with the positive support of $o(\vec{x})$, i.e. if $D$ is a set given by the positive support of a function $D(\vec{x})$, $o(\vec{x}) > 0$ iff $\vec{x} \in D$ should be true for as much $\vec{x}$ as possible.

If the set of $\vec{c}_k$ is given (see below), the training procedure of the Unit RBF is as follows:

Initialization: $d_1$ is the sign of $D(\vec{c}_1)$.

Step $k$: If $(k-1)$ steps were performed, then if

$$D(\vec{c}_k) \cdot \sum_{j=1}^{k-1} d_j e^{-j\|\vec{x}-\vec{c}_j\|^2} \leq 0$$

then

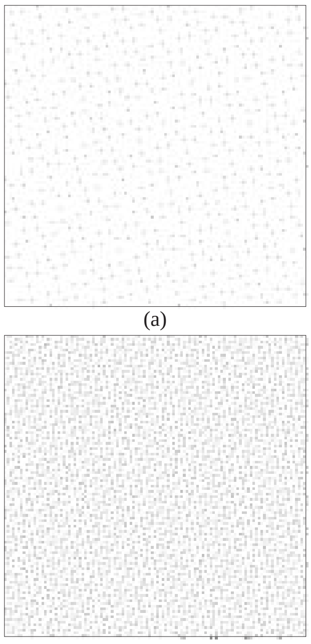
$$d_k = signD(\vec{c}_k)$$

else

$$d_k = 0.$$

That is, $d_k$ is 0 if the $(k-1)$-term function result produces the correct sign at $\vec{c}_k$; otherwise, $d_k$ is nonzero to correct the function.

In order to get a sequence of $\vec{c}_k$ positions, the so-called Linear Pixel Shuffling (LPS) is used. Choosing $\alpha$ as the positive real root of $\alpha^3 = \alpha^2 + 1$ ($\alpha \simeq 1.4655712$), and $\beta = \alpha(\alpha - 1)$, the $\vec{c}_k$ are for the two-dimensional case given with $(\{\alpha k\}, \{\beta k\})$. Here, $\{z\}$ denotes the fractional part of $z$. The LPS sequence samples the unit square in a manner illustrated in figure 8. It comprises a highly uniform sampling. In [**?**] and its reference, many examples are given for useful applications of LPS.

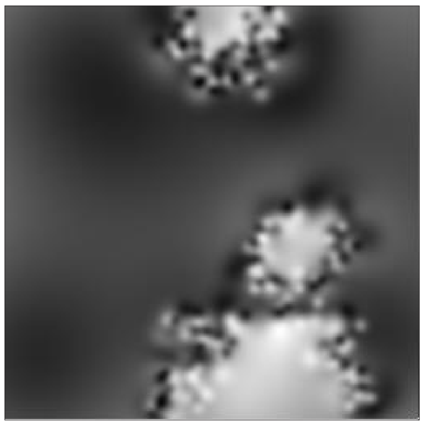

(a)

(b)

Figure 8: The first 500 (a) and 3000 (b) sampling points of the LPS procedure.

## 3.2 Unit RBF Approximation of 2D-Lookup matrix

The Unit RBF approach is directly applicable to the problem of approximating 2D-Lookup matrix images. Two images $I_1$ and $I_2$ are derived in the same manner as in the MBPN approach. Then, with each image a Unit RBF is trained. The plots of both functions obtained for the example of figure 7 (a) can be seen in figure 9. If both "recall" images are fusioned point-wise by the rules of table 2, the image in figure 10 is obtained. The approximation quality clearly outperforms that of the MBPN approach.


(a)

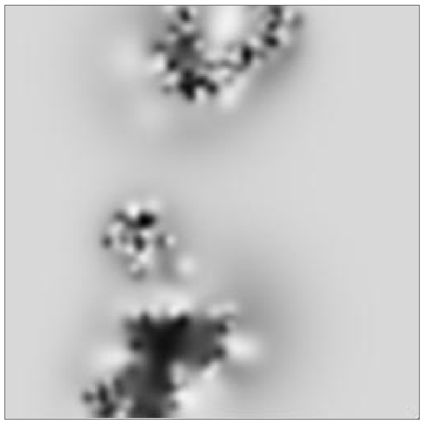

(b)

Figure 9: Images $I_1$ (a) and $I_2$ (b) approximated by the Unit RBF.

## 3.3 Fuzzy Interpretation

The texture model, which can be derived from the Unit RBF approximation, is a fuzzy one. For seeing that, consider the first nine terms of the expression, which lead to figure 9 (a):

$$\begin{gathered} o(x,y) = -e^{-4\|\vec{x}-(220,186)\|^2} + e^{-5\|\vec{x}-(83,105)\|^2} + \\ + e^{-6\|\vec{x}-(203,24)\|^2} - e^{-11\|\vec{x}-(31,129)\|^2} + \\ + e^{-13\|\vec{x}-(13,222)\|^2} - e^{-23\|\vec{x}-(181,177)\|^2} + \\ + e^{-24\|\vec{x}-(44,96)\|^2} + e^{-27\|\vec{x}-(146,108)\|^2} - \\ - e^{-42\|\vec{x}-(141,168)\|^2} + \ldots \end{gathered}$$

All Gaussian terms with $d_k = 1$ comprises fuzzy patches over the set of white pixels in the 2D-Lookup matrices. Thus, two-dimensional membership functions can be assigned, which are centered at $\vec{c}_k$ and the degree of membership of a position $\vec{x}$ to that membership function is given with $max(o(\vec{x}), 0)$. If we describe this membership by the linguistic term “$c_k$-like,” each rule has the form

$$if\,(op_1(x,y), op_2(x,y))\, is\, c_k - like,\, then\, 1$$

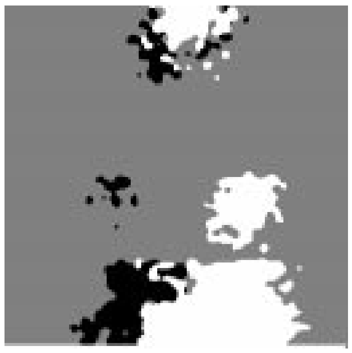


Figure 10: Approximation of the 2D-Lookup matrix of figure 7 (a) by the Unit RBF.

Table 2: Rules for fusion of the two Unit RBF images at position $(x,y)$.

| $rec_1$ | $rec_2$ | $rec$ |
|---|---|---|
| $> 127$ | $> 127$ | $max(rec_1, rec_2)$ |
| $> 127$ | $\leq 127$ | $rec_1$ |
| $\leq 127$ | $> 127$ | 127 |
| $\leq 127$ | $\leq 127$ | $min(rec_1, rec_2)$ |

Similar rules can be assigned from $I_2$ for the 0 case. Figure 11 shows the fuzzy landscape of the approximation in figure 9 (a), using the first 116 nonzero terms of $o(\vec{x})$.

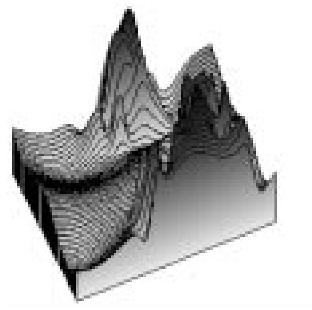


Figure 11: Surface plot of the approximation with 116 terms.

## 4 Summary

The work presented in this paper was motivated by the recently proposed 2D-Lookup framework for the evolutionary adaptation of texture filters. A special class of images, the 2D-Lookup matrices, plays an important role for the performance of the finally adapted texture filters. In order to improve the performance of the texture filters, the compactness of mostly black or white regions within the 2D-Lookup matrix image has to be increased. This task, which is somehow similar to a clustering of the 2D-Lookup matrices, can not be clustered by traditional approaches of cluster analysis due to the appearance of the prospective clusters within the images. Fuzzy region borders, intermingling of black and white dots and sparse distributions of black pixels over a large region are the unique features for the class of images, to which the 2D-Lookup matrices belongs. The task is considered as an approximation to a given 2D-Lookup matrix (as result of a texture filter generation performed in before-hand) by means of a suitable procedure.

Another important aspect is the possible derivation of texture models from the adapted filters, since the generation process of the texture filters is purely data-driven. If the selected procedure describes the 2D-Lookup matrices on a higher level than simply pixel-based, the whole 2D-Lookup algorithm can be specified by a set of assigned rules.

Since generalization ability is a requirement for a procedure, which is able to solve the task, neural networks were chosen as a means for approximating 2D-Lookup matrices. Two approaches were considered, one based on the multilayer backpropagation neural network (MBPN), and one based on the recently proposed Unit RBF network. Both approaches were shown to work good, but the Unit RBF approach is proven to be better able to represent important details of the 2D-Lookup matrices on different resolution scales. Moreover, the texture models, which can be derived from a Unit RBF are fuzzy ones, in contrary to the crisp nature of rules derived from a trained MBPN (in the case considered here, not in general). Thus, the Unit RBF is the more prominent approach to 2D-Lookup matrix approximation.

Once the rules are obtained, they can be used in many ways as follows.

- One possibility is to relate the rule parameters to global conditions of the initial texture problem, for which the filter was designed. For example, if the texture image becomes darker, the adapted filter would fail since the region of the 2D-Lookup matrix, which represents the textured background, has to be shifted to regions reflecting darker grayvalue pairs. If the 2D-Lookup matrix is just given pixel-wise, it is not obvious how to perform such a shift. If the 2D-Lookup matrix is given by rules, this shift can be performed by modifying the rule parameters.

- Several runs of the genetic algorithm used for adapting the 2D-Lookup framework on a given texture filtering problem yields different operation pairs. This way, the outcome of several runs can not be put together in order to attain a better texture filter. But if the texture filter is represented as a set of rules, the required mode of compatibility between different runs is achieved, and two ore more texture filters can be fused by simply unifying theirs set of rules into one set.

- Image processing steps applied on the initial texture image can be mapped onto processing steps of the rule base. E.g. texture synthesis by relaxation would become possible this way.

- The approximation procedure, if its result are reliable, can be incorporated into the fitness function computation, which is used for the evolutionary adaptation of the texture filters.

From this, the importance of 2D-Lookup matrix approximation, which was succesfully solved with the approaches presented in this paper, for the further development of the 2D-Lookup framework can be seen. Future work will concentrate on these issues.

## Acknowledgment

The research presented in this paper was supported by “Deutsche Forschungsgemeinschaft” (DFG), research project EPISTO Ni 473/1.