A Framework for the Evolutionary Generation of 2D-Lookup Based Texture Filters

Mario KÖPPEN, Martin TEUNIS, Bertram NICKOLAY

Fraunhofer Institute IPK-Berlin
Pascalstr. 8-9, 10587 Berlin, Germany
Phone: (++49)(0)30 390 06 200 Fax: (++49)(0)30 391 75 17
E-mail: {mario.koeppen|martin.teunis|bertram.nickolay}@ipk.fhg.de

Key Word: image processing, evolutionary algorithm, genetic algorithm, genetic programming, texture, texture filter, 2d-lookup, Nessy algorithm

Abstract

A framework is presented, which allows for the automated generation of texture filters by the exploitation of the 2D-Lookup algorithm and its optimization by means of evolutionary algorithms. To use the framework, one has to give an original image, containing the structural property-of-interest (e.g. a surface fault), and a binary image (goal image), wherein each position of the structural property-of-interest is labeled with the foreground color. Doing so, the framework becomes capable of evolving the specification of the 2D-Lookup algorithm, i.e. the two image processing operations and the 2D-Lookup matrix, towards a filter for the structural property-of-interest. Two versions of the framework will be considered, one using the genetic algorithm (GA), and one using the genetic programming (GP). For the GA approach, the filter generator is given access to a fixed set of image processing operations. The decoded bitstring specifies, which operations are selected out. Also, the decoded bitstring specifies the entries in the 2D-Lookup matrix. For the GP approach, the filter generator uses the expression tree of an individual to design two operations based on formal superoperators, which are referenced within the expression tree. The specification of the 2D-Lookup matrix is performed by a relaxation technique. It will be shown by some texture fault examples, that the GP approach performs better than the GA approach.

1 INTRODUCTION

Recently, the paradigm of visual routines [ULLMAN87] has found particular interest in the fields of active vision [BALA96] and visual agent design [JOHNSON94, KOEPPEN96]. Within this model, perception is considered as a distributed collection of task-specific, task-driven visual routines. Ways of doing an active control of what is processed by a vision system have been proposed.

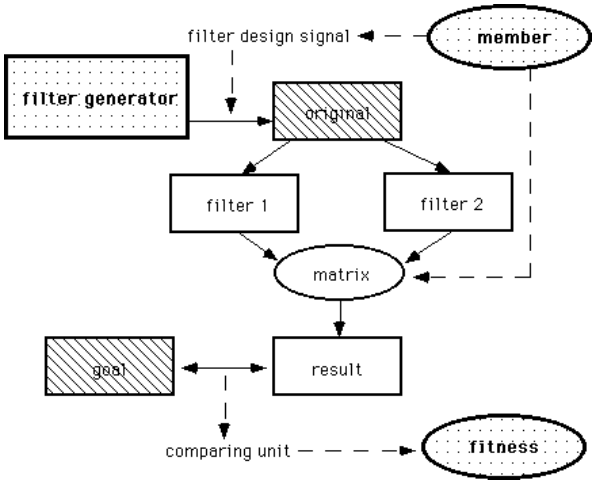


Fig. 1: The Framework for 2D-Lookup based texture filtering.

In [BALA96], the encoding through central control was considered as a possible way to craft visual routines. There, the feature classification framework was considered as a base, wherein features for facial landmarks detection tasks (e.g. eye detection) were extracted by a so-called hybrid GA. The advantage of crafting visual routines by employing a general framework, if compared with the genetic programming approach of [JOHNSON94], is the compact representation of well-known problem-solving

strategies, which had to be proven to be successful for given recognition tasks.
The emphasis of the work, which is presented here, is put on this "frameworking" for crafting visual routines. The 2D-Lookup algorithm, which is detailed in section 2.2., gives a much more flexible way to represent a visual routine than the approach of [BALA96], because the way of obtaining features is adapted by an evolutionary algorithm, too.
The paper is organized as follows. Section 2 describes the framework of 2D-Lookup based texture filter generation, in particular the 2D-Lookup algorithm, the role of the framework as a fitness evaluation unit, the measure for matching shapes used within the framework, and the GA and GP approach for evolving the specification of the framework in order to solve a given texture filtering task. Some results for images taken from "Textilfehler-Katalog," containing texture faults, are given in section 3. The paper concludes with a discussion of the results and a hint at future work in section 4.

2 DESCRIPTION OF THE FRAMEWORK

2.1 General Overview

The framework (see figure 1) is composed of (user-supplied) original image, filter generator, filter output images 1 & 2, result image, (user-supplied) goal image, 2D-Lookup matrix, comparing unit and filter design signal.
Each individual of the evolving population gives a unique specification of the 2D-Lookup algorithm. The so-specified algorithm is applied to the original image. Comparison of the result image and the goal image gives the fitness of the individual. This fitness measure is used for the operations of an evolutionary algorithm.

2.2 2D-Lookup Algorithm

The 2D-Lookup algorithm stems from mathematical morphology [SERRA82][SERRA88]. It was primarily intended for the segmentation of color images. However, the algorithm can be generalized to use two grayvalue images as well.
For starting off the 2D-Lookup algorithm, the two filter images 1 & 2, which are of equal size, need to be provided. This is achieved by the filter design signal, which is in control of the individuals of the evolving population. The filter design signal causes the filter generator to determine two image processing operations, which are applied to the original image. The 2D-Lookup algorithm goes over all positions of the filter images. For each position, the two pixel values at this position in filter images 1 & 2 are used as indices for looking-up the 2D-Lookup matrix. The matrix element, which is found there, is used as pixel value for this position of the result image. If the matrix is bi-valued (as for the goal image), the result image is a binary image.

2.3 Fitness Function

In order to compare the output image of the 2D-Lookup with the goal image, a quality function has to be designed. While the encoding of the framework is a rather straightforward task, the assignment of a good fitness function is not such as easy.

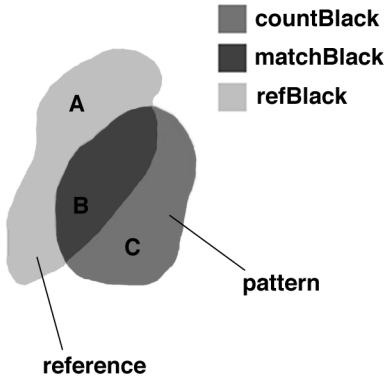


Fig. 2: Terms for fitness evaluation.

For obtaining a quality measure, the following terms are evaluated (see figure 2): **countBlack** (B+C) is the number of black pixels set in the result image, **matchBlack** is the number of black pixels of the result image, which are also black in the goal image (B), and **refBlack** is the number of black pixels of the goal image (A+B). Then, the following ratios can be computed:

r_1=**matchBlack**/**refBlack**,
r_2=1-(**countBlack**-**matchBlack**)/(*N*-**refBlack**) and
r_3=**matchBlack**/**countBlack**

with N be the total number of image pixels.
The multiple objective here is to increase these measures simultaneously. After performing some experiments with the framework, it was decided to use the following weighted sum for these three objectives:

$$f = 0.1\ r_1 + 0.5\ r_2 + 0.4\ r_3$$

This fitness measure has the following properties:

1. It counts better for subsets of the reference. Subsets obtain a fitness value of at least 0.9.
2. It counts better for subsets of the reference, which are supersets of other subsets of the reference.
3. A white image gives a fitness of 0.5, therewith refusing to assign a good fitness value to the empty subset of the reference.

These properties make this fitness measure useful for genetic search. A genetic search evolves its population towards the higher weighted objective first. In our case this means, that measure r_2, weighted with 0.5, is evolved first. With other words: the first subgoal of the genetic search is to allocate as many correct white positions as possible. Due to the weighting of 0.4 for the r_3-part, the search then tries to allocate correct black positions of the reference, while the correct white allocations persist in the pattern. Once the pattern is reduced to a subset of the reference, the only way to increase the fitness is to expand the subset towards the whole reference set. This begins, when the fitness exceeds a value of about 0.9.

2.4 GA Approach

In the GA approach, the filter generator is a table of pre-defined texture filters, which is referenced through the individual's filter design signal. As a result, the individual selects out two filter operations. In our case, this table was set up using 256 entries of image processing operations, which come from several categories (e.g. mathematical morphology, nonlinear filtering, ANOVA).

In order to use a GA, the framework has to be encoded into a chromosome. However, this encoding is a straightforward task. For the presented approach, a chromosome is represented by a bitstring of size 128. Bits 65 to 128 encode the 2D-Lookup matrix. The 256x256 square is downscaled to an 8x8 square. Every cell of the rescaled 8x8 matrix spans over a 32x32 grayvalue square of the 2D-Lookup matrix, setting them all to one or zero at once. 64 bits are needed to encode this downscaled matrix representation.

To keep the bitstring balanced, bits 1 to 64 represent the method choice, i.e. 32 bits must encode one of 256 possible operations. A redundant coding was used where the section of 32 bits was divided into eight subsections á four bits. The parities of all of these subsections of four bits, i.e. the number of even bits, are collected to give an eight bit binary number. This

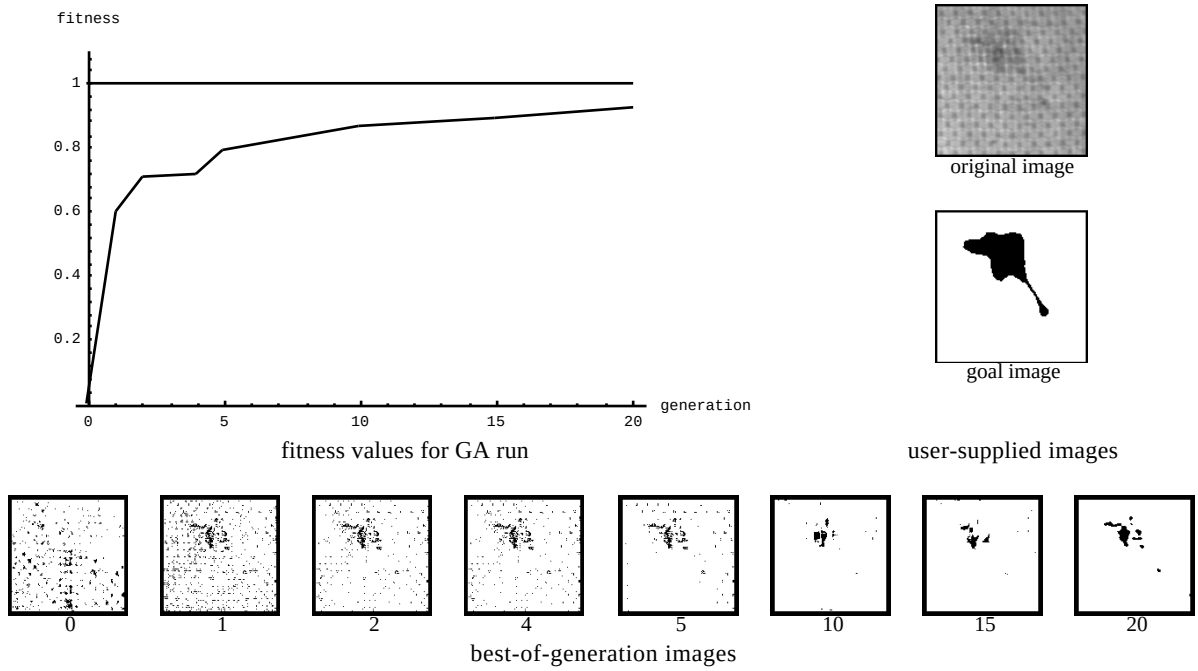


Fig. 3: Sample GA run for a texture fault example.

actually gives a number between 0 and 255, which is interpreted as operation index.
For choosing a genetic algorithm, it was decided to use the Nessy-algorithm [KOEPPEN97a,b] due to its algorithm can use these quality measures as fitness values. In this case, the evolutionary algorithm is referred to as genetic programming (GP). In the case of the GP approach, the problem is to generate image

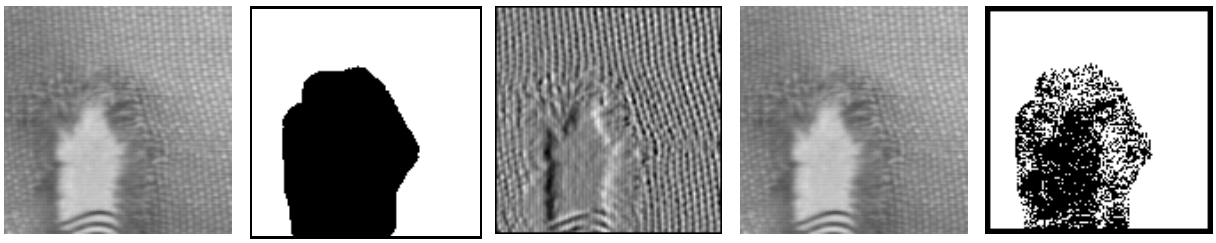


Fig. 4: Results for example 1, using GP approach.

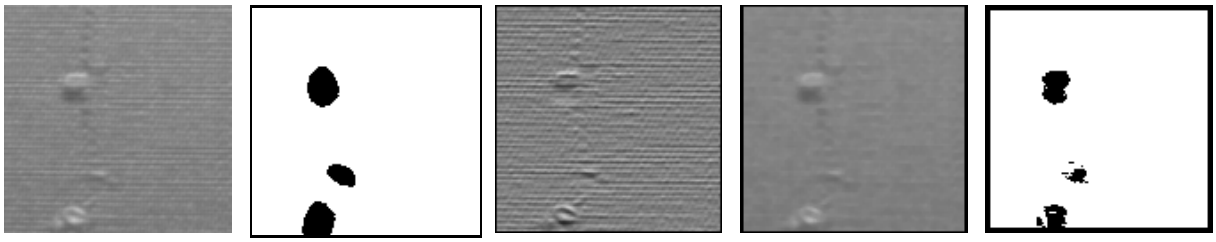


Fig. 5: Results for example 2, using GP approach.

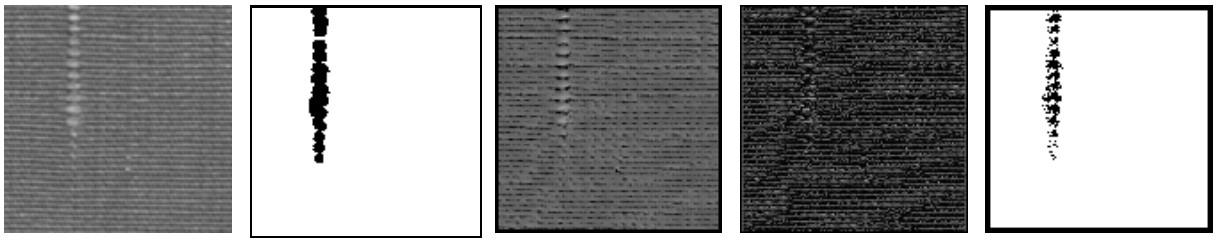


Fig. 6: Results for example 3, using GP approach.

excellent trade-off between exploitation and exploration in the genetic search. Figure 3 shows the evolution of a shape towards the goal image for a texture fault example.

2.5 GP Approach

2.5.1 Genetic Design of Filter Operations

One essential part of the framework are the two image processing operations. In order to generate them, genetic programming is used [KOZA92]. Genetic programming maintains a population of expression trees (shortly: trees). Every tree equals a program by its structure. "Performing" the tree may be used for evaluating a quality value for the suitability of the program represented by the tree. An evolutionary processing operations. Among the many choices for tree representations of image processing operations, the following approach is used here: the terminals of each tree are parameter structures with a mode entry. The mode entry specifies a superoperator, the specification of which is given with the parameter settings. The aritry of the root level of each tree is two in order to generate two image processing operators.
The trees are kept small in depth (maximum depth is three) and uses only node functions out of a small set of functions (minus, minimum, maximum, square, evaluate).
The following superoperators were used: Translation, Convolution, Ordered Weighted Averaging (OWA) [YAGER88], Texture numbers and Simple Fuzzy Integral [GROSSERT96, SUGENO74].

2.5.2 Setting of the 2D-Lookup Matrix

A relaxation-based technique is presented here for the GP case, which checks every matrix position within a 256x256 grid for a possible increase in the fitness value of the result image, if it is set black. If $G=(g_1,g_2)$ is a pair of grayvalues, then be M(G) the set of all positions (x,y) in the operations images, where operation image 1 has grayvalue g_1 and operation image 2 has grayvalue g_2. Then, the fitness of M(G), which is considered as a binary image, is computed as given above, and if this fitness value exceeds 0.88, i.e. if the majority of these positions is in the black part of the goal image, then position G is set to black in the 2D-Lookup matrix. Shortly spoken, the 2D-Lookup matrix is the union of all G, where *fitness*(M(G)) is above 0.88.

This relaxation-based technique works very good in the context of 2D-Lookup matrix specification. But the generalization ability of the deduced filters may decrease due to errors in the goal image. The designed filter can depend on random grayvalue patterns occurring within the image. But, as can be seen from figure 8 in the next section, the 2D-Lookup matrix regions of black positions remains compact, or border a compact white region, if the textural property-of-interest of the goal image is separable.

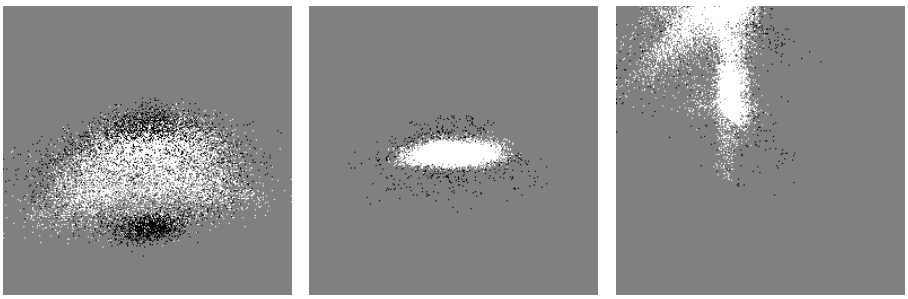
An attempt to bypass the possible problem of over-specialization is the use of a self-organizing feature map neural network (SOM) for the segmentation of the 2D-Histogram into 64 clusters. By co-evolution, the setting of each cluster to be black or white can be adapted to obtain better-matching result images. The main problem here is the computational effort for separately training a SOM and co-evolving the corresponding matrix specification for the fitness evaluation of each individual.

example 1 example 2 example 3

Fig. 7: Results for the same examples as in figures 4-6, using GA approach.

3 RESULTS

To learn about the framework's properties, textile images were used which were taken from the "Textil-Fehlerkatalog" of the IPK Berlin*. Three examples will be shown here. In order to test the frameworking approach, subimages were cut from the whole images, containing the fault and its neighboring texture.



example 1 example 2 example 3

Fig. 8: 2D-Lookup matrixes for the examples 1 to 4, obtained with the GP approach.

Results for these four examples will be given in the following subsections.

The results for the GP approach are presented first. The results for a run for each problem, including both filter images, are given in the figures 4 to 6. For each

* The "Textilfehler-Katalog" is accessible via *http://vision.fhg.de/ipk/tfk.*

problem, a population of 30 parent trees and 80 children was used, the run went on until the genetic diversity of the population approached nil. Usually, this was the case after performing about ten generations.

The corresponding results for the GA approach are given in figure 7. Here, the Nessy-algorithm was configured as follows[†]: 10 solution layer neurons, 64 generation layer neurons and one output neuron. Learning rate was 1.0. The Nessy-network was iterated for 30 cycles.

Obviously, the GP approach performs better than the GA approach. Especially, better results are yielded for examples 1 and 2. However, for example 1 this is not really a better filter. Figure 8 shows the 2D-Lookup matrixes, which were obtained by the GP approach. Black and white dots indicate positions, which are set to black and white respectively in the result images, gray dots indicate grayvalue pairs, which do not appear in both filter images. It is of major importance, that black and white regions in the matrixes are separable. This is the case for the examples 2 and 3. There, black dots are bordering compact white regions. But, as can be seen from figure 8, too, this is not the case for example 1. This filter is not expected to generalize well.

4 DISCUSSION AND FUTURE WORK

A framework was presented, which allows for the design of texture filters for fault detection (two class problem). The framework is based on the 2D-Lookup algorithm, where two filter output images are used as input.

The approach was applied to three texture problems and the performance of the framework was shortly discussed. The results, obtained without "human intervention," are ready-to-use texture filters. Also, they can be tuned in order to obtain even more better results, or combined in a superposed inspection system. The following experiences were made during the test runs:

- The framework is able to design texture filters with good or very good performance.
- The goal image should match the fault region as good as possible.
- Bordering regions should be neglected for fitness evaluation.
- The framework is able to design filters for the detection of non-compact fault regions and fault regions with varying appearance.
- The designed filters may be subjected to further improvements by the designer.

Current work focuses on several improvements of the whole architecture, especially on the inclusion of rescaling and rescanning into the designed filter operations, and an evaluation of the 2D-Lookup matrix by neural networks in order to get a comprehensive solution for a given texture filtering problem.

Reference

[BALA96] Bala, J., Huang, H., Vafaie, H., Wechsler, H., "Visual Routine for Eye Detection Using Hybrid Genetic Architectures", *Proc. ICPR'96*, Vienna, Austria, Vol. III, pp. 606-610, 1996

[GROSSERT96] Grossert, S., Koeppen, M., Nickolay, B., "A new approach to fuzzy morphology based on fuzzy integral and its application in image processing," *Proc. ICPR'96*, Vienna, Austria, Vol. II, pp. 625-630, 1996

[JOHNSON94] Johnson, M.P., Maes, P., Darrell, T., "Evolving Visual Routines", in: *Brooks, R., Maes, P., ed., "Artificial Life IV: Proc. of the 4th Int. Workshop on the Synthesis and Simulation of Living Systems*", MIT Press, Cambridge , MA, July 1994

[KOEPPEN96] Köppen, M., Nickolay, B., "Design of image exploring agent using genetic programming," *Proc. IIZUKA'96*, Iizuka, Japan, pp. 549-552, 1996

[KOEPPEN97a] Köppen, M., Teunis, M., Nickolay, B., "A neural network that uses evolutionary learning," *Proc. IEEE ICEC'97*, Indianapolis, Indiana, pp. 635-639, 1997

[KOEPPEN97b] Köppen, M., Teunis, M., Nickolay, B., "Nessy - an evolutionary learnig neural network," *Proc. SOCO'97*, Nîmes, France, pp. 243-248, 1997

[KOZA92] Koza, J., "Genetic programming - on the programming of computers by means of natural selection," MIT Press, Cambridge, London, 1992

[SERRA82] Serra, J., "Image analysis and mathematical morphology," Academic Press, London, 1982

[SERRA88] Serra, J., ed., "Image analysis and mathematical morphology. Volume 2: theoretical advances," Academic Press, London, 1988

[SUGENO74] Sugeno, M., "Theory of fuzzy integral and its applications," Tokyo Inst. of Technology, Ph.D. thesis, 1974

[ULLMAN87] Ullman, S., "Visual routines," in: *Fischler, M.A., Firschein, O., ed., "Readings in Computer Vision,"* pp. 298-327, 1987

[YAGER88] Yager, R.R., "On ordered weighted averaging aggregation operators in multi-criteria decision making," *IEEE Trans. on SMC* 18, pp. 183-190, 1988

[†] The Nessy-algorithm is implemented as a neural network. Solution, generation and output layer resemble input, hidden and output layer of the conventional multilayer perzeptron, but its neuron counts resemble the number of parents and children of a conventional GA.