11 A framework for the adaptation of image operators

Mario Köppen and Raul Vicente-Garcia

11.1. Introduction

The automated tuning of image processing operations is an important task for the improvement of the robustness, reliability, and versatility of image processing systems. Nearly every approach in this field is based on the classical image processing chain (IPC), which consists of a sequence of single image processing operations steps that are designed independently. Mostly notable steps here are the image acquisition, the computation of features and their classification. Other steps that might extend the processing chain are image enhancement, region-of-interest specification or image segmentation before feature computation, feature selection or feature transformation before classification, and semantic processing of images classes or object detection following the classification. Among many textbooks about this field, see especially [15] for an excellent introduction and motivation.

The versatility of the IPC scheme is usually achieved by means of a training procedure (see Figure 11.1). Given a training set (either labeled data supplied by the user, or unlabeled ones in the so-called unsupervised learning mode), the training scheme may modify some of the internal settings of the steps in the IPC in order to achieve the best mapping function from images to classes. A remarkable issue here is that the single steps in the processing flow consider their input usually as immutable: the feature computation does not modify the image acquisition procedure, the classification does not influence the manner in which the features were computed. The training overcomes this drawback by being given some influence on these settings, like modification of some parameters of the feature computation, or modification of internal parameters of the classification. However, from the fact that each step was designed independently, it follows that training can always be broken into parts, which tune a single step only.

From this description, the most important drawback of the IPC approach becomes obvious: the IPC as a whole cannot perform better than its worst configured part.

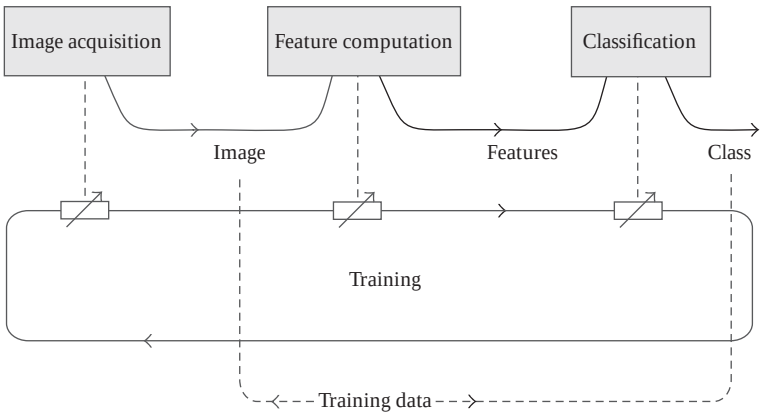


FIGURE 11.1. Typical processing flow for the training of an image processing system.

However, another problem with the IPC approach is not that obvious: each free parameter of an IPC gives an additional dimension of the search space of the optimization problem that corresponds to the training. Thus, the search space seems to become quite large. However, a simple quantitative investigation of the case gives, that the search space, if given in terms of the represented image (and not the IPC free parameters) is apparently greater, since it is given by a mapping. The number of possible mappings of n variables with m possible values onto k values is $k^{(m^n)}$. In fact, the percentage of possible image processing operations that can be represented by an IPC with respect to the total number of *all* possible image processing operations is nearly zero, even if the domain of the latter ones is highly restricted.

When soft computing techniques like genetic algorithms, genetic programming or neural networks are to be applied to the design of image processing operations, the question of how image processing operations can be adequately represented for this purpose becomes very important.

The evolutionarily designed primary visual system of higher mammals gives an important hint on this issue. Here, for example, the model of the Boundary Contour System/Feature Contour System [5] introduced two independent pathways in the cortical processing of (at least) static images, which are perceived by the retina and cortical cells.

The point of interest in the context mentioned afore is the use of several pathways instead of a single one (i.e., an IPC). Within the final fusion of the processing results of each pathway, a mapping can be included, which dramatically increases the number of representable operations. If the mapping can be specified independently, it can be expected that the task of image processing operation tuning can be solved much more effectively.

Once having the general idea of such n-dimensional frameworks, the main body of this chapter is considering a realization of such a framework. It comes out that a basic generic algorithm, originating in the image processing discipline of

mathematical morphology, the so-called 2D lookup, provides all that is needed to design such a framework.

For adapting the framework to the solution of a given image processing task, genetic programming (GP) will be used. The task here is to evolve the operations ("pathways") that give the input for the mapping in the framework. Using GP for adapting the IPC has been studied a few times in the past. The seminal work of Tackett [20] used GP to derive efficient feature computations, and the application was the recognition of targets. Harris and Buxton proposed the use of GP for deriving edge detectors in 1D signals [6], while Poli studied the more general use of GP in several image processing tasks like image enhancement, image filtering, and feature classification [14]. A specialization to the detection of objects or regions-of-interest was presented by Bhanu and Lin [2, 3]. All these approaches followed the general idea to synthesize more complex operators from simple ones and showed the efficiency of such approaches. Using rules instead was considered by Stanhope and Daida in [18]. A recent work by Lam and Ciesielski introduced the computation of translation invariant features that are evolved by GP and evaluated by a clustering procedure [12].

The motivation to use a 2D-Lookup algorithm as internal mapping in an IPC was firstly inspired by the successful application of a 2D-Lookup for the segmentation of background texture in images of bank checks [4]. Later on, a refined version was presented by Köppen et al. in [8, 9]. The application of the presented framework to an industrial collagen-sheet inspection system can be found in [13], and its application to texture detection (with the intention to select appropriate image region for digital watermarking) can be found in [7].

In this contribution, the 2D frameworks based on this general idea and means for its extension will be presented. In Section 11.2 the concept of an n-dimensional framework will be presented and discussed, followed by Section 11.3 that introduces the 2D-Lookup algorithm for realizing such a framework. Details of the framework are presented in Section 11.4, and the possible extensions are discussed in Section 11.5. After the provision of some results for the application of the framework in Section 11.6, the chapter concludes with a short summary and the reference.

11.2. Multidimensional frameworks

In this section, a rough estimation should be made about the dimensionalities of the search problem involved in an optimization approach to IPC configuration. The basic assumption is that the result of the IPC should resemble a given goal image as good as possible, or that some properties are fulfilled by the result image. From this, the configuration of an IPC comes out to be an optimization problem.

Consider Figure 11.2, where a simple unit IPC is given. It is assumed that the computations are restricted within a 3×3 neighborhood at each pixel. All grayvalues at image pixel locations are assumed to be values between 0 and 255. The result of the operation should be a binary one, that is, the computations give a value of either 0 or 1. Then, such a unit IPC can be considered as a mapping

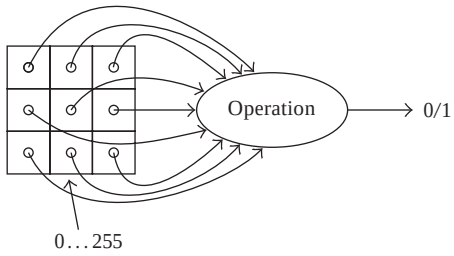


FIGURE 11.2. The unit IPC which maps nine grayvalues onto two.

$f : \{0,\ldots,255\}^9 \to \{0,1\}$ from nine grayvalues onto the set of binary values $\{0,1\}$. If there is a quality function q that assigns a quality value to each mapping f, a mapping f_{opt} is searched, for which its quality value becomes optimal. However, since for the mapping f there are 256^9 function values to specify, each of which can either be 0 or 1, there are $2^{(256^9)}$ possible mappings for the unit IPC. This is the number of elements of the search space, too.

As an example, consider the class of convolution operations. A weighted mask is given by the mapping of an index set M (the mask) into the set of weights, like

$$M_w = \begin{array}{|c|c|c|}\hline w_{(-1,-1)} & w_{(0,-1)} & w_{(1,-1)} \\ \hline w_{(-1,0)} & w_{(0,0)} & w_{(1,0)} \\ \hline w_{(-1,1)} & w_{(0,1)} & w_{(1,1)} \\ \hline \end{array} \tag{11.1}$$

and assuming here that $w_{ij} \in \{0,\ldots,255\}$. Then, convolving the image I at position (x, y) with the weighted mask M_w can be written as

$$R(x,y) = \sum_{(i,j)\in M} w(i,j)I(x+i, y+j). \tag{11.2}$$

By thresholding the result with a value ϑ, a binary image is obtained. For this unit IPC with nine parameters, there are 256^9 possible choices for the parameters. Compared with the number of search space elements $2^{(256^9)}$, this is only a marginal amount of representable operations. The number p of parameters with a domain $\{0,\ldots,255\}$ that would be needed to cover the search space completely is $256^9/8$. This follows from equating $2^{(256^9)} = 256^p = 2^{8p}$.

A serious problem arises from this consideration. If adaptive techniques like soft computing methods should be applied to such an image processing problem, the search space is much too big to be covered by the search method. There seems to be no way to represent an arbitrary mapping of the kind of mappings used in image processing operations.

The number of represented operations could be dramatically increased, if a mapping would be involved into the IPC operations. This leads to the definition

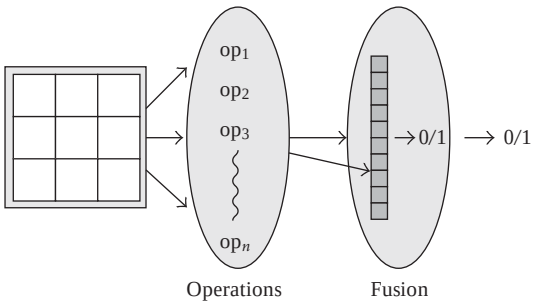


FIGURE 11.3. An n-dimensional framework, which decomposes an IPC into n operations performed in parallel and a final fusion of the operation results. The fusion can be based on a mapping, thus heavily increasing the number of representable operations.

of n-dimensional frameworks. An n-dimensional framework is a decomposition of the processing flow into n parallel parts op_1 to op_n and a final fusion procedure (see Figure 11.3). Each single operation is applied onto the original image, and then, the n result images are fused by an appropriate algorithm. If the fusion is specified by a mapping of n values out of a set of m values each onto the set $\{0, 1\}$, the framework, as seen from the "outside," serves as a unit IPC of the kind given above. There are $2^{(m^n)}$ mappings specified. If m is set to 256 and n to 9, we exactly meet the requirements of the unit IPC.

This is the key idea of n-dimensional frameworks: they allow for its adaptation as a whole by adapting the parameters of n operations (each of which could be an IPC itself), thereby sampling the search space to a much more larger degree as can be achieved by setitngs a number of internal parameters only.

The question is, of course, if there is a fusion algorithm that really supports the adaptation of image processing operators. The answer is positive, and in the next section, the 2D-Lookup algorithm will be identified as such a fusion procedure.

11.3. 2D-Lookup algorithm

The 2D-Lookup algorithm stems from mathematical morphology [16, 17]. It was primarily intended for the segmentation of color images. However, the algorithm can be generalized for using on grayvalue images as well.

For applying the 2D-Lookup algorithm, two input images g_1 and g_2 of same size and number of bits per pixel are required. These two images can be the result of applying two image operators onto a single input image, or by extracting color channels from a color representation of an input image. The nature of the image operators or channel selections itself does not matter for the application of the 2D lookup. The other component of the algorithm is a matrix of dimension $N_g \times N_g$ (with N_g being the number of different grayvalues in the two input images) and having entries from a set of labels, with preference to the same grayvalue range as the input images.

```
for x=0 to img_width-1 do
begin
    for y=0 to img_height-1 do
    begin
        g1 = g1(x, y)
        g2 = g2(x, y)
        out (x, y) = l (g1, g2)
    end y
end x
```

Algorithm 11.1

The 2D-Lookup algorithm goes over all common positions of the two-operation images. For each position, the two pixel values at this position in the images g_1 and g_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, the resulting image is a binary image.

Let I_1 and I_2 be two grayvalue images, defined by their image functions g_1 and g_2 over their common domain $P \subseteq N \times N$:

$$\begin{aligned} g_1 &: P \longrightarrow \{0, \ldots, g_{\max}\}, \\ g_2 &: P \longrightarrow \{0, \ldots, g_{\max}\}. \end{aligned} \tag{11.3}$$

The 2D-Lookup matrix is also given as an image function l, but its domain is not the set of all image positions but the set of tupels of possible grayvalue pairs $\{0, \ldots, g_{\max}\} \times \{0, \ldots, g_{\max}\}$,

$$l : \{0, \ldots, g_{\max}\} \times \{0, \ldots, g_{\max}\} \longrightarrow S \subseteq \{0, \ldots, g_{\max}\}. \tag{11.4}$$

Then, the resulting image function is given by

$$\begin{aligned} r &: P \longrightarrow S, \\ r(x, y) &= l(g_1(x, y), g_2(x, y)). \end{aligned} \tag{11.5}$$

In standard applications, every grayvalue is coded by eight bit, resulting in a maximum grayvalue of 255. Also, the domain of the image function is a rectangle. In this case, the 2D-Lookup is performed by the pseudocode shown in Algorithm 11.1.

To give a simple example for the 2D-Lookup procedure, $g_{\max} = 3$ is assumed in the following. Let

$$g_1 : \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline 0 & 3 & 3 \\ \hline \end{array}, \qquad g_2 : \begin{array}{|c|c|c|} \hline 2 & 3 & 1 \\ \hline 2 & 3 & 2 \\ \hline \end{array} \tag{11.6}$$

be the two input images and let the 2D-Lookup matrix be given by

$$l: \begin{array}{|c|c|c|c|c|} \hline g_2 \backslash g_1 & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 2 & 2 \\ \hline 2 & 1 & 2 & 3 & 3 \\ \hline 3 & 2 & 3 & 3 & 2 \\ \hline \end{array} \tag{11.7}$$

Then, the resulting image is

$$r: \begin{array}{|c|c|c|} \hline l(0,2) & l(1,3) & l(2,1) \\ \hline l(0,2) & l(3,3) & l(3,2) \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 1 & 3 & 2 \\ \hline 1 & 2 & 3 \\ \hline \end{array} \tag{11.8}$$

In the following, the 2D-Lookup matrix will only contain the two entries: Black (0) and White (1).

A typical base for the matrix can be the so-called 2D histogram. Using the former notation, the 2D histogram of two images g_1 and g_2 is a mapping

$$H(g_a, g_b) = \sum_{(x,y)\in P} \delta(g_1(x,y), g_a)\,\delta(g_2(x,y), g_b) \tag{11.9}$$

with $\delta(a,b)$ being 1 for $a = b$ and 0 otherwise. The entry of H at position of a grayvalue pair (g_a, g_b) contains the number of pixel positions (x, y) where image g_1 has grayvalue g_a and image g_2 has grayvalue g_b. Using a normalization factor (like the maximum value in H), the 2D histogram can be given as an image and used as a 2D-Lookup matrix.

Figure 11.4 illustrates the relations between 2D histogram and 2D lookup by means of the Lena image. The subimage to the middle right is the 2D histogram of the Lena image that was obtained from the red and green channels of the Lena image (taken as grayvalue images). There are some obvious clusters in this histogram that give rise to a labeling as shown in the figure. The five binary images were obtained by using a 2D-Lookup matrix image each, having all positions of the segment number i set to black, and white otherwise. So, it can be seen that the binary image generated by setting all points of label 3 to black (lower-left subimage) basically covers the mirror structure in the image background. Since the projection of label 3 into both axis directions of the label image is crossing label 2, it is not possible to extract this mirror structure from the red or green channel image of the Lena image alone. More precisely: if one tries to extract the grayvalue range spanned by the positions of the mirror structure, in either case (red or green) also

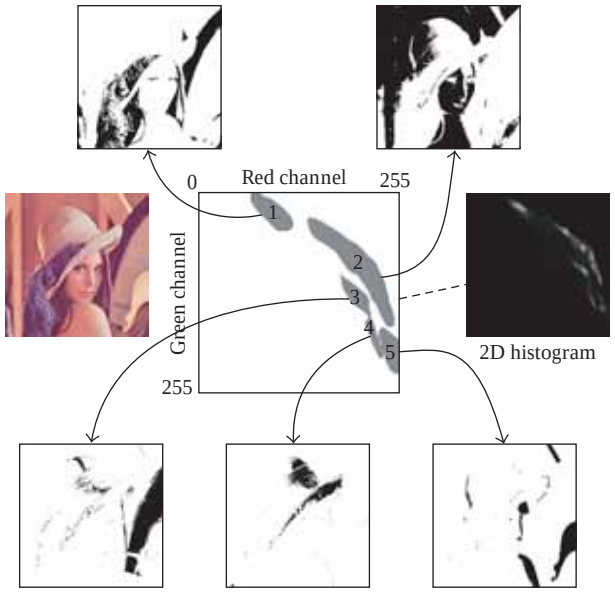


FIGURE 11.4. Various segmentations of Lena image based on labeling of the 2D histogram of red and green channels.

some other structures are always selected as well that do not belong to the mirror structure. The projections of the 2D histogram gives a clear indication for this. The 2D-Lookup algorithm is able to separate the mirror structure directly. From this simple example, the potential of the 2D-Lookup algorithm for segmentation tasks can be seen.

11.4. 2D-Lookup-based framework

The 2D-Lookup-based framework (see Figure 11.5) is composed of (user-given) original image, filter generator, operation images 1 and 2, result image, (user-supplied) goal image, 2D-Lookup matrix, comparing unit, and filter generation signal.

The framework can be thought of as being composed of three (overlapping) layers.

(1) The instruction layer, which consists of the user-supplied parts of the framework: original image and goal image.
(2) The algorithm layer performs the actual 2D-Lookup, once all of its components (original image, operation 1, operation 2, and 2D-Lookup matrix) are given.
(3) The adaptation layer contains all adaptable components of the framework (operation 1, operation 2, 2D-Lookup matrix) and additional components for performing the adaptation (comparison unit, filter generator).

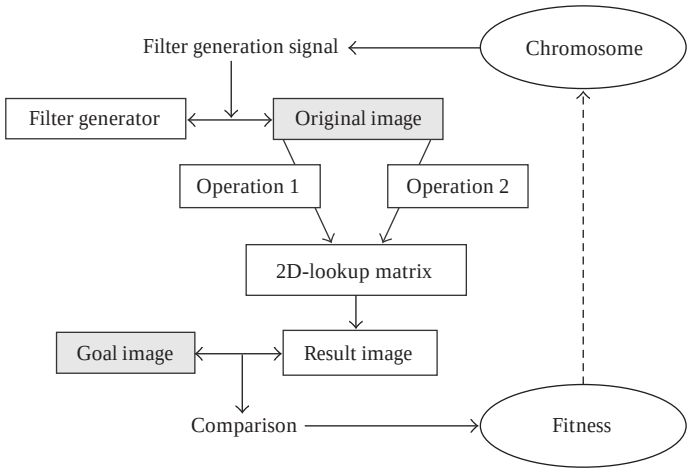


FIGURE 11.5. 2D-Lookup-based framework: overview.

For the instruction layer, the user interface has been designed as simple as possible. The user instructs the framework by manually drawing a (binary) goal image that provides the wanted segmentation of the original image into foreground and background. In this image, all pixels of the background segment are set to White and all pixels of the foreground (e.g., the location of a texture fault, or the handwriting on a textured bankcheck background) are set to Black. No special texture model has to be known by the user. There are no further requirements for the goal image.

For making the framework the subject of an evolutionary adaptation, several aspects have to be considered in more detail.

(1) Fitness function: assuming the operation images and the 2D-Lookup matrix to be given, the result of 2D lookup has to be compared with the user-supplied binary goal image (indicated as "comparison" in Figure 11.5). This is achieved by a fitness function that measures the degree of spatial correspondence between two binary images.

(2) Derivation of 2D-Lookup matrix: once the two-operation images are known, it can be expected that the 2D-Lookup matrix can be adapted in order to give the best fitness to the goal image as result of the 2D lookup. Later on (see Section 11.4.2) we will provide exactly such a procedure, making the 2D-Lookup matrix a function of the operation images 1 and 2 and the goal image alone, and without the need of any further adaptation.

(3) Having the fitness function and the method to derive the 2D-Lookup matrix, it only remains to specify the two-operation images. Here, the task can be handled by an evolutionary procedure. In summary, we decided to use the representation of operations as expression trees, and using genetic programming in order to derive optimal image operations.

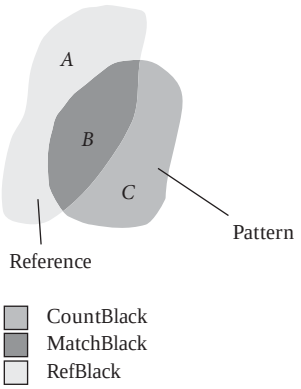


FIGURE 11.6. Terms for fitness evaluation.

In the following sections, these necessary specifications of the framework will be presented.

11.4.1. Fitness function

In order to compare the output image of the 2D Lookup with the goal image, a quality function has to be designed for the comparison of two binary images. First, the definition of this fitness function will be given, then it will be discussed.

Consider Figure 11.6, where two sets are shown, the reference set of the goal image and the pattern set of the result image. The reference set of the goal image is the set of all black pixels in the goal image that is given by the user. The pattern set of the result image is the set of all black pixels of the result image.

Therein, countBlack is the number of black pixels in the result image ($B + C$), 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:

$$r_1 = \frac{\text{matchBlack}}{\text{refBlack}}, \tag{11.10}$$

where r_1 is the sensitivity, or amount of reference pixels matched by the pattern,

$$r_2 = 1.0 - \frac{\text{countBlack} - \text{matchBlack}}{N - \text{refBlack}}, \tag{11.11}$$

where r_2 is the specificity, or amount of correct white pixels set in the result image (N is the total number of image pixels), and

$$r_3 = \frac{\text{matchBlack}}{\text{countBlack}}, \tag{11.12}$$

where r_3 is the positive predictivity, or percentage of matching pixels of the result image. In case of empty reference or pattern sets (white images), r_1 and r_3 are set to 0, respectively.

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 of these three objectives as fitness measure:

$$f_{\text{reference}}(\text{pattern}) = 0.1r_1 + 0.5r_2 + 0.4r_3. \tag{11.13}$$

This fitness measure has the following properties.

(1) It counts better for patterns that are subsets of the reference. Subsets obtain a fitness value of at least 0.9, since in this case specificity r_2 and positive predictivity r_3 both take the value 1.
(2) It counts better for patterns that are subsets of the reference, and which are supersets of other patterns that are also subsets of the reference (supersets of patterns have a larger value for sensitivity r_1 than the pattern, and again, $r_2 = r_3 = 1$).
(3) A white image as pattern gives a fitness of 0.5 (only specificity $r_2 \neq 0$), therewith refusing to assign a good fitness value to the empty subset of the reference.

These properties make this fitness measure useful for heuristic search procedures. Initially, higher fitness values can be obtained by increasing the higher weighted objective first. In our case this means that specificity r_2, weighted with 0.5, is improved first. In other words, the first subgoal of the search could be to allocate as many correct white positions as possible. Due to the slightly smaller weighting of 0.4 for the positive predictivity r_3, the search then could continue to allocate also correct black positions of the reference, while the correct white allocations persist in the pattern. Once, by this exploration, 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, and exploitation starts.

11.4.2. Deriving a 2D-Lookup matrix

It was already noted that the specification of a 2D-Lookup matrix can be done without the need for a separate adaptation. To make this more clearly, we recall the dependencies between the parts of the framework. Using the notations *in* for the original image, goal for the goal image, op_1 and op_2 for the two-operation images after applying the operators o_1 and o_2 to in, res for the result image and lt_2 for the 2D-Lookup matrix, the 2D-Lookup LU_2 can be formally written as

$$\text{res} = \text{LU}_2(o_1(in), o_2(in), lt_2) = \text{LU}_2(\text{op}_1, \text{op}_2, lt_2), \tag{11.14}$$

and thus it can be expected that there is also an operation or algorithm REL to specify lt_2 from the other terms in (11.14):

$$lt_2 = \text{REL}\,(\text{op}_1, \text{op}_2, \text{res}) = \text{REL}\,(\text{op}_1, \text{op}_2, \text{goal}). \tag{11.15}$$

The replacement of res with goal expresses the fact that the primary intention of the adaptation is to have the relation res = goal fulfilled as good as possible.

For specifying REL, we consider a family of simple 2D-Lookup matrices, where all positions but one position (a, b) are set to White (1), and the remaining position (a, b) is set to Black (0). Then, the 2D lookup will give a resulting image with all positions (x, y) set to Black, for which operation o_1 yielded pixel value a at (x, y) in op_1 and operation o_2 yielded pixel value b at (x, y) in op_2. Usually, there will be only a few black pixels within the result image. Now, we compute the fitness measure in (11.13) taking the set of black positions in the result image as the pattern set, and the black pixels of the goal image as reference set. As it was remarked in the former section, the fitness measure will give values above 0.9, if the pattern set of black pixels lies completely within the reference set, even if this set contains only one pixel. So, a simple criterion can be derived for setting a pixel to Black or White in the 2D-Lookup matrix.

Let $l_{(a,b)}$ be a two-dimensional matrix constituted by setting only the position at (a, b) to Black (0) and all others to White (1), and let $\text{res}_{(a,b)}$ be the result of the 2D lookup with the operation images op_1 and op_2 and this matrix $l_{(a,b)}$. Then we use for REL:

$$lt_2(a, b) = \begin{cases} \text{Black (0)} & \text{if } f_{\text{goal}}(\text{res}_{(a,b)}) > 0.88, \\ \text{White (1)} & \text{otherwise.} \end{cases} \tag{11.16}$$

It can be seen that this procedure REL only requires the operation images op_1 and op_2, and the goal image goal. In case there are no black pixels in $\text{res}_{(a,b)}$ at all, $lt_2(a, b)$ is set to Gray (0.5), which stands for positions within the 2D-Lookup matrix, whose pixel value pairs do never occur within the operation images op_1 and op_2 at the same location. The value 0.88 has been chosen instead of 0.9 to tolerate a few black pixels in $\text{res}_{(a,b)}$ to be out of the goal set.

Figure 11.7 shows the result of the algorithm REL for the derivation of a suitable 2D-Lookup matrix for two example operation images. This procedure, which resembles a relaxation procedure, gives a quasioptimal 2D-Lookup matrix for given operation images op_1 and op_2.

By this specification, the algorithm looks rather time consuming. However, by doing some book keeping, and using the fact that only points will be set to Black in the 2D-Lookup matrix, for which the corresponding grayvalue pairs in the operation images have at least one position that is also set black in the goal image, the processing time can be remarkably reduced. In the pseudocode shown in Algorithm 11.2, the 2D-Lookup matrix is derived as a grayvalue image `lt` of size 256×256 and with Black positions set to grayvalue 0, White psoitions to 255, and Gray positions to 127.

FIGURE 11.7. In the middle, the result of applying algorithm REL to two-operation images (left) and given goal image (bottom right) can be seen, and top right is the result of the application of this 2D-Lookup matrix to the two-operation images.

```
Algorithm REL:
Input: op1, op2 - operation images; goal - goal image
Output: lt - grayvalue image of size 256x256
_ _ _
Init:
- set all pixels of lt to 127
- compute refBlack from goal

Algorithm:
for all pixel positions (i, j) with goal (i, j) = 0
begin
    g1 := op1(i, j), g2 := op2(i, j)
    if lt (g1, g2) = 127 then
      countBlack := 0; matchBlack := 0;
      for all pixel positions (k, l) in op1
        if op1(k, l) = g1 and op2(k, l) = g2 then
            countBlack++;
            if goal (k, l) = 0 then matchBlack++; end if
        endif
       compute f = f (countBlack, matchBlack, refBlack)
        if f >= 0.88 then lt (i, j) := 0;
          else lt (i, j) := 255;
        endif
      end for all (k, l)
    endif
end for all (i, j)
```

ALGORITHM 11.2

The following section describes the manner by which the two operations needed are derived by an individual of a GP.

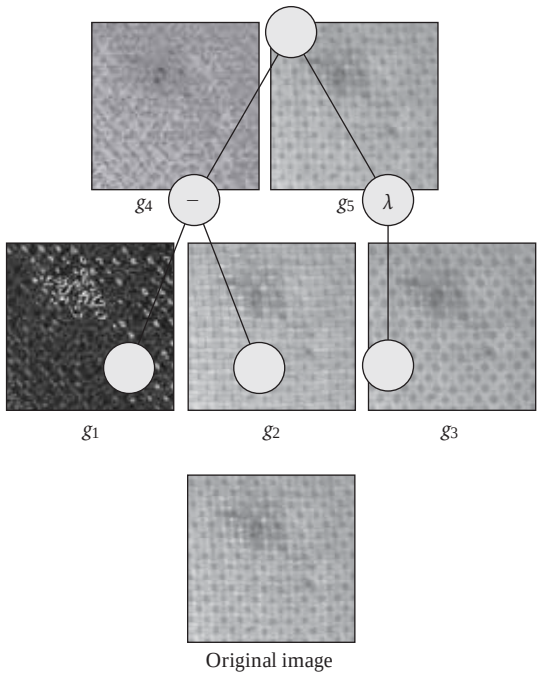


FIGURE 11.8. Example for the intermediate images that were generated for an operator tree with operation images 1 and 2 on the top row.

11.4.3. Generic operator design

The operator selection is based on a tree-like representation of the image processing operations. Each node in the tree refers to a generic image processing operation out of the set squaring, square root, move, ordered weighted averaging (OWA [21]), fuzzy T-norm, and fuzzy integral [19] for nodes of arity one, and pixelwise addition, subtraction, multiplication, and T-Norm for nodes of arity two (higher-arity nodes are not used). The parameters of such a node operation are given by a *parameter structure resource*, with the same structure for all nodes. This parameter structure resource indicates an offset vector (for move operation), a weighted mask (for OWA, T-norm and fuzzy integral), a weighting vector (for OWA) and some flags and mode values. See [8] for more details on the operation specifications.

In the framework presented here, the operator selection is performed two times, to get the operation images, from which the 2D-Lookup matrix is derived. In all cases, the operation trees and the parameter structures are randomly initialized, and adapted later on by genetic programming [10, 11] as optimization procedure to gain high fitness values.

The structuring of image processing operations by the trees has been chosen in this manner for the following reasons.

(i) The random selection of operations, which are represented by such trees, can be done in a user-driven manner that favors well-known image processing operations. Thus, a tree could be made more likely to represent operations as dilation, erosion, closing, opening, morphological gradients, Sobel operator, statistical operators, Gaussian filtering, shadow images and so forth.
(ii) The represented operations are unlikely to give unwanted operation images, which are completely white or black.
(iii) The employed operators are local, that is, for each position of the result image, the computed value is a function of the grayvalues of a bounded domain (containing the same position) of the original image.

The maximum arity of a node is set to two. Also, maximum tree depth was restricted to five. This was set in order to allow for the maintenance of the obtained trees, for example, for manually improving the designed filters by removing redundant branches. Processing time is kept low, too (but in the present version of the framework, processing time does not go into the fitness function itself!).

Figure 11.8 shows a typical tree constructed in this manner and applied to an input image with a fault structure on textured background (bottom subimage). Figure 11.9 gives some operation images obtained from the same original image by different randomly constructed and configured trees. These images demonstrate the variabilty of the generated operations, each of which enhances or surpresses different image substructures, and none of which gives a trivial image operation.

11.5. Framework extensions

In this section, we will introduce two optional extensions of the framework. The next section is devoted to the aspect of obtaining filters with higher generalization ability, and then, an optional preprocessing module based on 2D lookup with the 2D histogram is presented.

11.5.1. Reconstruction of the 2D-Lookup matrix

An important question is the generalization ability of the designed filters. While they were designed for one and only one input-goal image pair, it may not be obvious how the filter performs, if they are applied to the same situation presented by 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 11.10 shows some 2D-Lookup matrices, which were the result of applying the framework to various texture fault examples (the texture faults themselves are of minor interest for the following). By considering these matrices, the following can be noted.

(i) Some matrices seem to be separable, that is, the textured background is represented by a group of compact white regions. The fault appearance (the black dots) are well separated from these regions.

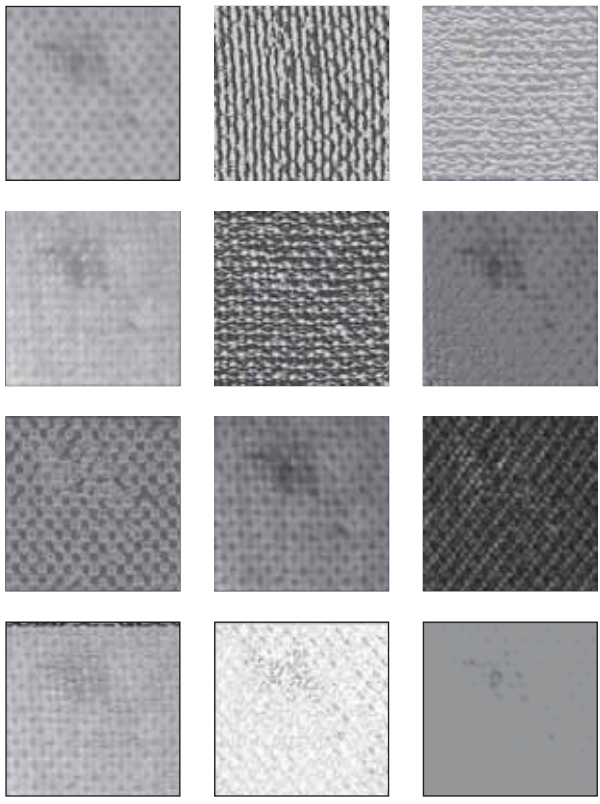


FIGURE 11.9. Result images of a random initialization of a population of image processing operators.

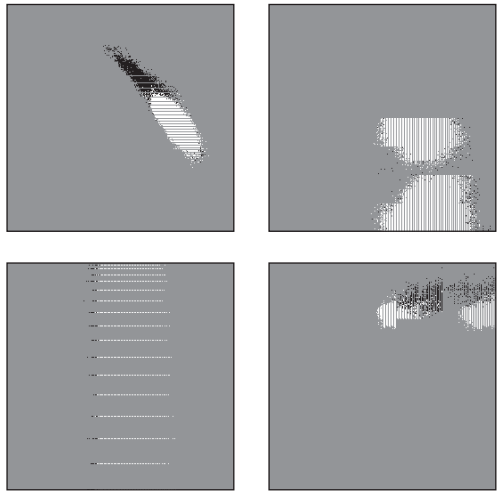


FIGURE 11.10. Example for 2D-Lookup matrices generated by the 2D-Lookup framework.

(ii) The gray parts of the matrices represent positions, for which no special rule could be assigned, since the corresponding pair of grayvalues was not present in the operation images at the same position.

(iii) There are noncompact, noisy regions, where black and white dots are completely admixed. This seems to be a combined effect. The grayvalues around some position, which was indicated as foreground in the goal image, could be similar (but not equal) to grayvalues around positions that were indicated as background in both operation images. This is a conflict in the assignment, as given by the goal image, and the rapid changes between black and white for closeby positions reflect such an ambiguity of an optimal assignment. The conclusion is that these filters are not able to achieve a good separation between image foreground and background.

(iv) Some matrices are separable by a horizontal or vertical straight line (e.g., lower left example of Figure 11.10). This means that the 2D-Lookup algorithm can be simplified to thresholding op_2 or op_1, respectively.

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.

Neural networks would give a suitable procedure for the segmentation of the matrix, since they attain generalization from data. In the following, an approach based on the use of the unit radial basis function network (Unit-RBF), as proposed in [1] will be given. The Unit-RBF approach is directly applicable to the problem of approximating 2D-Lookup matrix images. The derived 2D-Lookup matrix was decomposed into a black part and a white part. Since there are often fewer foregound pixels in the goal images than background pixels, the black part image was further processed by morphological closing operation (to join some isolated black dots into a single segment). Then, the Unit-RBF approach was used to reconstruct both images, rec_1 from the black part, rec_2 from the white part, and both images were pixel-wise fused into a single image by using the rules given in Table 11.1. Note that the entry 127 stands for undecidable positions, where no evidence can be obtained from the framework adaptation.

Figures 11.11 and 11.12 give some examples for the replacement of the derived 2D-Lookup matrix with the reconstructed one. It can be seen that the differences between the results are not very large, and that the reconstructed matrices have a much simpler structure.

TABLE 11.1. 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	≤ 127	rec_1
≤ 127	>127	127
≤ 127	≤ 127	$\min(rec_1, rec_2)$

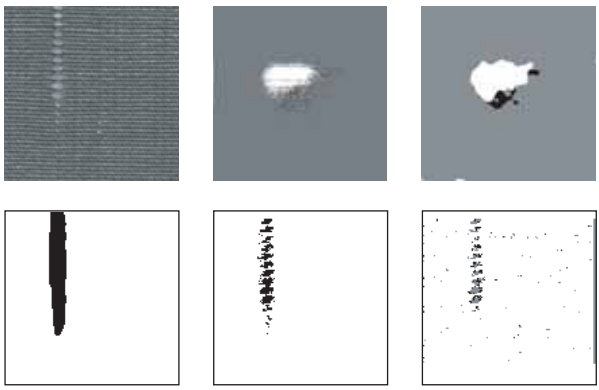


FIGURE 11.11. Reconstruction of 2D-Lookup matrix. Upper row shows the original image, the 2D-Lookup matrix as it was adapted by the 2D-Lookup-based framework, and its reconstructed version. The lower row shows the goal image, the result of 2D lookup using adapted 2D-Lookup matrix, and the result using the reconstructed matrix.

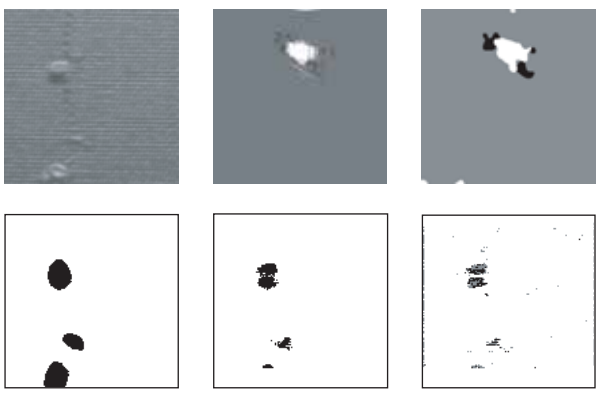


FIGURE 11.12. Another example for 2D-Lookup matrix reconstruction. The lower row shows the goal image, the result of 2D lookup using adapted 2D-Lookup matrix, and the result using the reconstructed matrix.

11.5.2. Optional preprocessing module

The framework has shown a good performance on a broad range of filtering tasks. Filtering here means the separation of image foreground and background.

(a) (b) (c)

FIGURE 11.13. Low-contrast stroke processing (a) shows a stroke on a collagen sheet, (b) shows a detail enlarged, (c) gives the output of best 2D-Lookup adaptation, demonstrating the poor performance of this algorithm in this case.

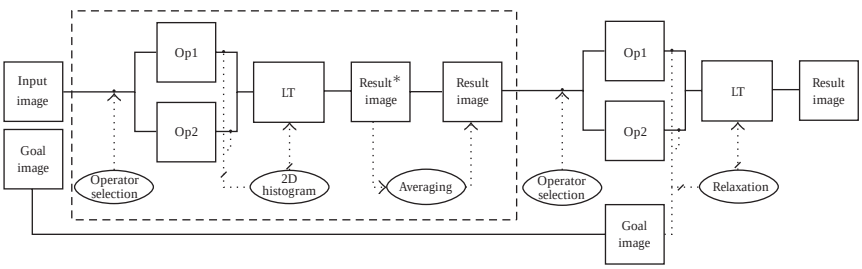


FIGURE 11.14. Extended framework for 2D-Lookup adaptation, with optional 2D histogram lookup preprocessing module.

However, it also showed some falacities. So, a rather poor performance on foreground appearance with very low contrast against the background was noted. Figure 11.13 gives an example. The stroke-like fault structure in Figure 11.13(a) (a scratch in a collagen sheet), clearly visible for a human, becomes nearly "invisible" while getting enlarged (see Figure 11.13(b)). Figure 11.13(c) shows the result of 2D-Lookup adaptation, revealing only some random dot locations of the fault.

Analyzing this problem, it came out that the framework employs local image processing operators only. Local processing here means that the domain of the image processing operations is bounded to a spatial neighborhood of each pixel. Detection of a stroke as the one in Figure 11.13 cannot be achieved by such a local processing.

This subsection presents an approach to solving such problems, by introducing an extension of the 2D-Lookup framework, referred to as 2D histogram lookup procedure. It is also based on the 2D lookup, but uses the normalized 2D histogram as 2D-Lookup matrix. The main advantage of this approach is that the mapping assigns a low grayvalue to positions with grayvalue pairs that appear infrequently in the operation images (there are fewer corresponding entries in the 2D histogram). This increases the contrast of "untypical" image structures against the background, and thus simplifies the following 2D-Lookup adaptation task.

Figure 11.14 shows the general framework for 2D-Lookup adaptation, with the optional 2D histogram lookup extension. The components will be described in

the following sections. As can be seen, the extension has nearly the same structure as the framework without extension, only the specification of the lookup matrix LT is different.

The 2D histogram lookup is the 2D-Lookup algorithm with using the normalized 2D histogram as lookup matrix (see end of Section 11.3).

For using the 2D histogram as lookup matrix, its entries have to be scaled into the range of feasible grayvalues. For usual goal image sizes, entries in the 2D histogram are seldom larger than the maximum grayvalue, so the entries themselves can be used as lookup matrix values, bounded at 255. Also, a contrast improvement by linearization is reasonable, for gaining better contrast in the result image. Linearization is the operation of making the sum-histogram of grayvalues stepwise linear. It replaces each grayvalue in the image with the relative number of grayvalues equal or below this grayvalue. Hereby, the large number of zero entries in the 2D histogram is neglected, thus starting linearization at grayvalue 1.

While gaining higher contrast by linearization, the number of different grayvalues now in the result image becomes reduced. This gives images with a cluttered appearance of the 2D-Lookup matrices in the following 2D-Lookup adaptation step. This effect can be reduced by using a Gaussian smoothing operator of size 3 on the linearized image.

The framework for 2D-Lookup adaptation was extended in order to enable the processing of low-contrast foregrounds. Figure 11.15 gives a complete example for such an adapted filtering. This example demonstrates several things.

(i) The final result image in the lower left is much more similar to the goal image than in Figure 11.13(c).

(ii) The result of the 2D histogram lookup preprocessing step has increased the contrast of the stroke against the background. This allows for the following 2D-Lookup adaptation step to achieve that better performance.

(iii) The operation trees are not “smart” in the sense that parts of them may do not have much influence on the operation result of the full tree. So, the left wing of the operation 1 of the 2D lookup (the tree left below in Figure 11.15) basically produces a gray image, from which a shifted version of the preprocessed image is subtracted. Such parts can be removed in a manual redesign phase of the filters after adaptation (especially when they involve computationally more expensive node operations like the fuzzy integral).

(iv) The 2D lookup for this example is basically a lookup with the cooccurrence matrix of the preprocessed image.

(v) The essential part of this filter lies in the operation 2 of the preprocessing (upper right tree): the 2D histogram, with its line-like structure, forks for some higher grayvalue pairs, thus stimulating a separation of image parts into foreground and background.

Figure 11.16 shows another example, without the intermediate results. This is a so-called “vibrating knife” fault that may occur in collagen slicing. The extended framework shows a good performance here as well.

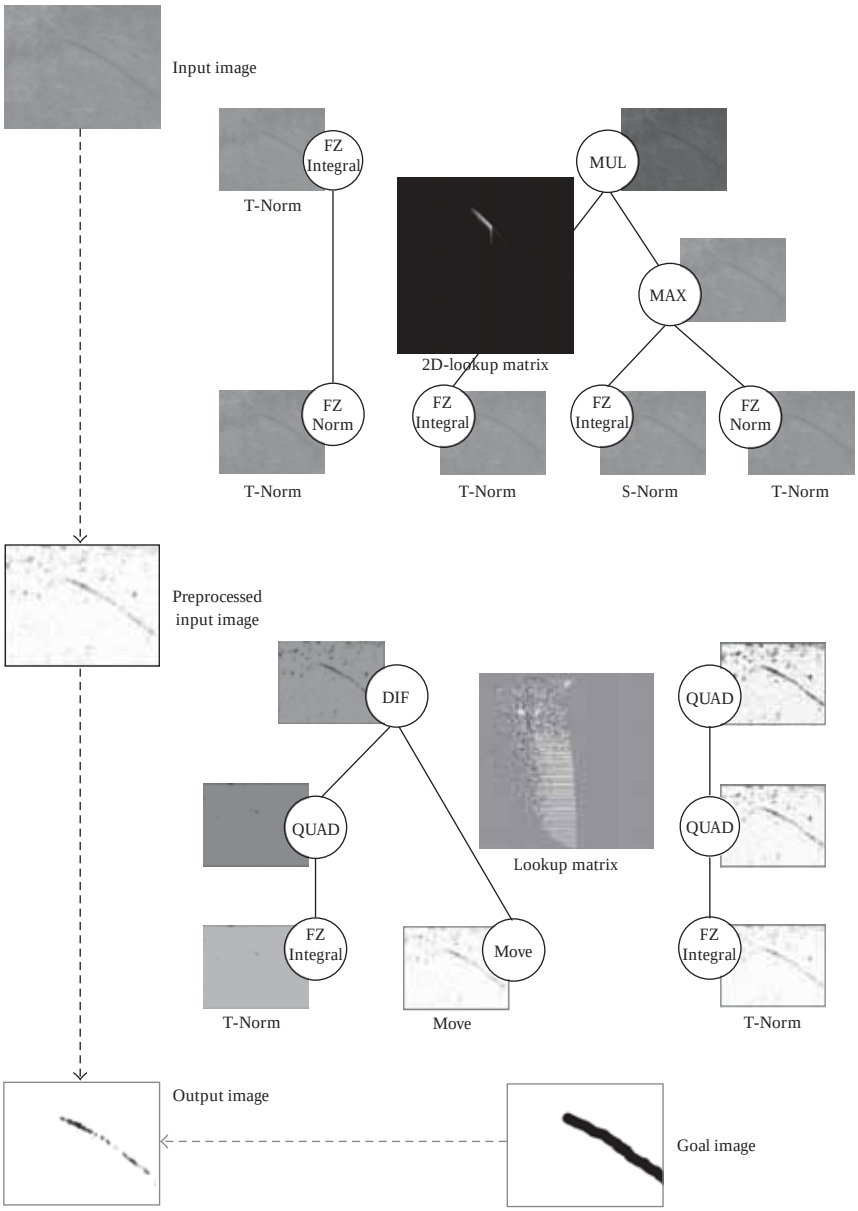
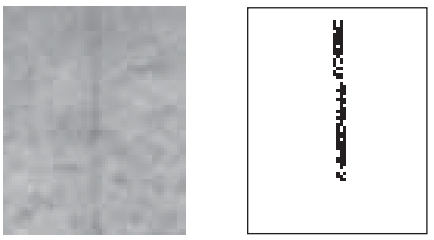


FIGURE 11.15. Complete example for the extended framework adapted to the low-contrast stroke in Figure 11.1.

11.6. Some results

In this section, some results of the application of the presented framework to some texture analysis problem are presented. For the genetic programming, setting was



(a) (b)

FIGURE 11.16. Result of the extended framework for vibrating-knife fault.

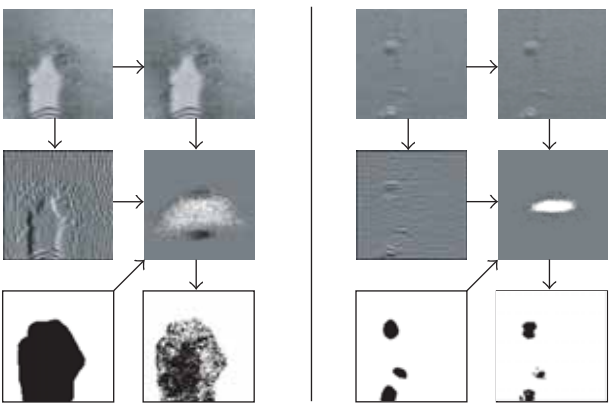


FIGURE 11.17. Application of the framework to textile fault problems.

standard: 40 trees were used in each generation, and standard tournament crossover was the only genetic operator used in each generation to produce 80 children nodes. The size of initial trees was restricted to have at least three nodes and to have not more than 9 nodes. The evolution ran until there was no diversity anymore in the population, which usually happened after about ten generations.

In Figures 11.17–11.20 the subimage order of each block is: upper-left subimage is the original image, lower-left is the user-given goal image. Right and below the original image are the two-operation images that are the result of the application of the evolutionary adapted trees to the original image, and that are the input operands for 2D-Lookup algorithm. The arrows from the operation images point to the used 2D-Lookup matrix (also indicated as a function of the two-operation images and the goal image), and lower right is the obtained result image.

11.7. Summary

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.

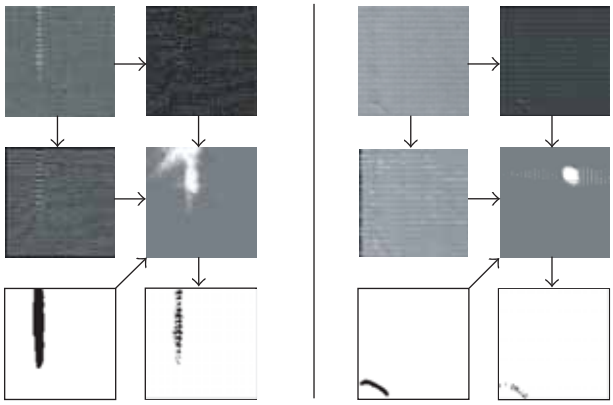


FIGURE 11.18. Application of the framework to further textile fault problems.

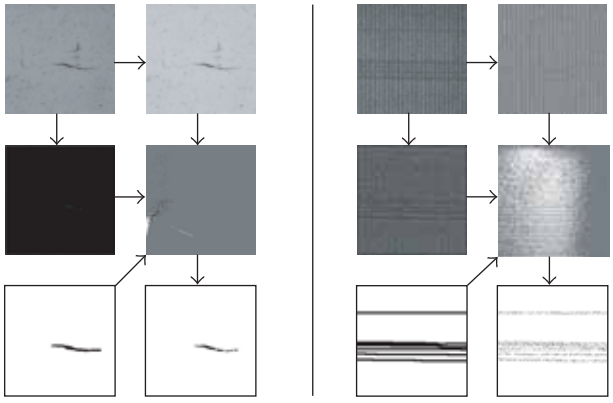
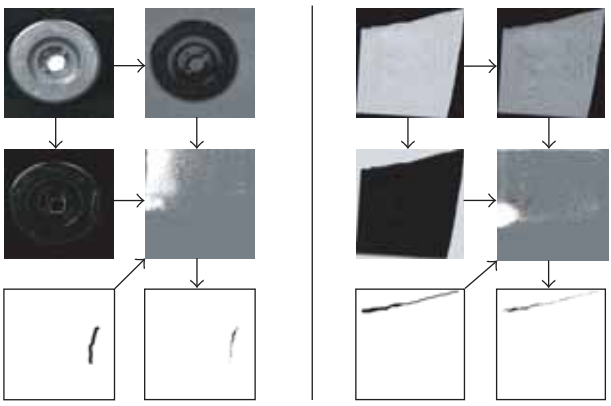


FIGURE 11.19. Application of the framework to floor pattern fault problems.

The approach can be applied to a large class of texture analysis problems. 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 are our experiences during the use of the system.

(i) The framework was able to design texture filters with good or very good performance.
(ii) The goal image matched the fault region quite satisfactorily.
(iii) Bordering regions should be neglected for fitness evaluation.
(iv) The framework was able to design filters for the detection of noncompact fault regions and fault regions with varying appearance.
(v) The designed filters may be subjected to further improvements by the user.



(a) (b)

FIGURE 11.20. (a) application of the framework to cast fracture problem and (b) extraction of sheering area on torned paper piece.

Improvements of the whole architecture were considered as well: one is based on an evaluation of the 2D-Lookup matrix by neural networks in order to get a more comprehensive solution for a given texture filtering problem, the other for extending the application scope to low-contrast texture fault processing, that is, faults which are hard to separate from the background texture. The second extension of the framework is a two-stage one, based on 2D histogram lookup and consecuting 2D-Lookup adaptation.

Bibliography

[1] P. G. Anderson, "The unit RBF network: experiments and preliminary results," in *Proceedings of the International ICSC/IFAC Symposium on Neural Computation (NC '98)*, M. Heiss, Ed., pp. 292–297, International Computer Science Conventions, Vienna, Austria, September 1998.

[2] B. Bhanu and Y. Lin, "Learning composite operators for object detection," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pp. 1003–1010, New York, NY, USA, July 2002.

[3] B. Bhanu and Y. Lin, "Object detection in multi-modal images using genetic programming," *Applied Soft Computing Journal*, vol. 4, no. 2, pp. 175–201, 2004.

[4] K. Franke and M. Köppen, "Towards an universal approach to background removal in images of bankchecks," in *Proceedings of the 6th International Workshop on Frontiers in Handwriting Recognition (IWFHR '98)*, pp. 55–66, Taejon, Korea, August 1998.

[5] S. Grossberg, "A solution of the figure-ground problem for biological vision," *Neural Networks*, vol. 6, no. 4, pp. 463–483, 1993.

[6] C. Harris and B. Buxton, "Evolving edge detectors with genetic programming," in *Proceedings of the1st Annual Conference on Genetic Programming (GP '96)*, pp. 309–314, MIT Press, Cambridge, Mass, USA, July 1996.

[7] M. Köppen and X. Liu, "Texture detection by genetic programming," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '01)*, vol. 2, pp. 867–872, Seoul, South Korea, May 2001.

[8] M. Köppen and B. Nickolay, "Genetic programming based texture filtering framework," in *Pattern Recognition in Soft Computing Paradigm*, N. R. Pal, Ed., vol. 2 of *FLSI Soft Computing Series*, pp. 275–304, World Scientific, Singapore, 2001.

[9] M. Köppen, A. Zentner, and B. Nickolay, "Deriving rules from evolutionary adapted texture filters by neural networks," in *Proceedings of IEEE International Conference on Fuzzy Systems (FUZZY '99)*, vol. 2, pp. 785–790, Seoul, South Korea, 1999.
[10] J. R. Koza, *Genetic Programming—On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass, USA, 1992.
[11] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge, Mass, USA, 1994.
[12] B. T. Lam and V. Ciesielski, "Applying genetic programming to learn spatial differences between textures using a translation invariant representation," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '05)*, vol. 3, pp. 2202–2209, Edinburgh, UK, September 2005.
[13] M. Köppen, A. Soria-Frisch, and T. Sy, "Using soft computing for a prototype collagen plate inspection system," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '03)*, vol. 4, pp. 2844–2850, Canberra, Australia, December 2003.
[14] R. Poli, "Genetic programming for feature detection and image segmentation," in *Evolutionary Computation, AISB Workshop*, T. C. Forgarty, Ed., pp. 110–125, Springer, Brighton, UK, April 1996.
[15] D. G. Stork, R. O. Duda, and P. E. Hart, *Pattern Classification*, John Wiley & Sons, New York, NY, USA, 2nd edition, 2000.
[16] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, UK, 1982.
[17] J. Serra, *Image Analysis and Mathematical Morphology 2: Theoretical Advances*, Academic Press, London, UK, 1988.
[18] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, pp. 735–744, Springer, San Diego, Calif, USA, March 1998.
[19] M. Sugeno, *Fuzzy Control*, Nikkan Kogyo Shimbun-sha, Tokyo, Japan, 1988.
[20] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, S. Forrest, Ed., pp. 303–309, Morgan Kaufmann, Urbana-Champaign, Ill, USA, June 1993.
[21] R. R. Yager, "On ordered weighted averaging aggregation operators in multi-criteria decision making," *IEEE Transaction on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183–190, 1988.

Mario Köppen: Department of Artificial Intelligence, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi, Fukuoka 820-8502, Japan
Email: mkoeppen@pluto.ai.kyutech.ac.jp

Raul Vicente-Garcia: Department Automation Technologies, Fraunhofer-Institute for Production Systems and Design Technology, Pascalstr. 8-9, 10587 Berlin, Germany
Email: raul.vicente@ipk.fraunhofer.de