2D-Histogram Lookup for Low-contrast Fault Processing

Mario Köppen, Raul Vicente Garcia, Xiufen Liu and Bertram Nickolay
Fraunhofer IPK, Department Pattern Recognition
Pascalstr. 8-9, 10587 Berlin, Germany
email: {mario.koeppen|raul|xiufen.liu|nickolay}@ipk.fhg.de

Abstract. This paper presents a framework for low-contrast texture fault processing based on 2D-Histogram lookup. 2D-Histogram lookup is a variant of 2D-Lookup operation. For 2D-Lookup, in two differently processed versions of the same image, grayvalue pairs from the same location in both images are replaced by the corresponding entry in a given two-dimensional matrix. The two operations and the matrix have to be provided for full algorithm specification. For 2D-Histogram lookup, the matrix is derived from the 2D-Histogram of both processed images. The main advantage of using the 2D-Histogram is the darkening of rarely occurring structures in the image, while highly probable image structures becomes bright. The so-processed images are then given to a 2D-Lookup procedure for automatic filter generation. For low-contrast texture faults, i.e. faults which are hard to separate from the background texture, the approach shows better performance in fault region detection than the approach of 2D-Lookup adaptation without 2D-Histogram lookup. For handwriting separation from textured background, no achievement was obtained.

1 Introduction

Recently, a framework was presented for the automated generation of texture filters [1]. The framework was based on the adaptation of 2D-Lookup algorithm application of Mathematical Morphology to a user-given task (see section 2.2 for the definition of this algorithm). The task includes the provision of the iconic description of the image foreground structure (texture fault, handwriting on textured background) by means of a so-called goal image, and the original image.

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

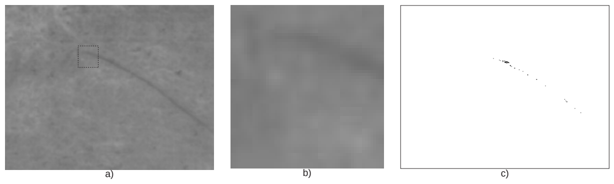

Figure 1: Low contrast stroke processing: (a) shows a stroke on a collagen sheet, (b) a detail enlarged. (c) gives the output of best 2D-Lookup adaptation, demonstrating the poor performance of this algorithm in this case.

However, it also showed some falacities. So, a rather poor performance on foreground appearance with very low contrast against the background was noted. Fig. 1 gives an example. The stroke-like fault structure in subfigure (a) (a scratch in a collagen sheet), clearly visible for a human, becomes nearly ”invisible” while getting enlarged (subfigure (b)). Subfigure (c) shows the result of 2D-Lookup adaptation, revealing only some random dots of the fault.

Analyzing this problem, it came out that the framework only employed a purely local image processing. 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 fig. 1 can not be achieved by such a local processing.

This paper presents an approach to solving such problems, by introducing an extension of the 2D-Lookup framework, referred to as 2D-Histogram lookup procedure. This extension can be considered a pre-processing with the whole image as operation domain. It is also based on the 2D-Lookup, but uses the normalized 2D-Histogram as lookup matrix. This gives higher contrasting of locally ”untypical” image structures, thus simplifying the consecuting 2D-Lookup adaptation task.

Section 2 describes the now extended framework and its components, while section 3 gives some examples for the approach. The paper concludes with a summary and the reference.

2 General framework

2.1 Overview

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

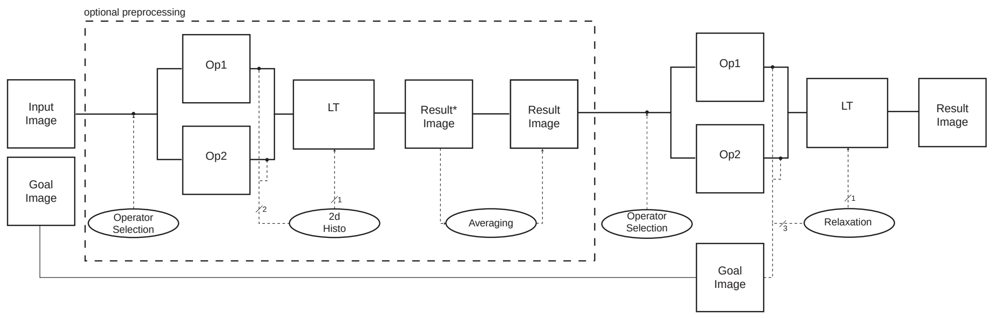

Figure 2: Extended framework for 2D-Lookup adaptation, with optional 2D-Histogram lookup pre-processing module.

The 2D-Lookup algorithm is specified by three components: image operator 1, image operator 2, giving operation images Op1 and Op2, and the lookup matrix LT. Both image operators can be choosen arbitrarily, hence the high adaptive potential of this algorithm. The choice of the operators is performed by the operator selection module.

For choosing an appropriate lookup matrix, the framework without extension and the extension follow different strategies. While in the framework the lookup matrix is set up in accordance with the goal image, in order to achieve maximum similarity of 2D-Lookup result and goal image (measured with a fitness function), in the extension the lookup matrix is taken from the 2D-Histogram of two processed versions of the input image. However, in this case, a post-processing of the result image by an averaging operation is necessary, as indicated by the naming ”result*.”

2.2 2D-Lookup

The 2D-Lookup algorithm stems from mathematical morphology [4], [5]. It was primarily intended for the segmentation of color image channels. However, the algorithm can be generalized to be used for grayvalue images as well.

For starting off the 2D-Lookup algorithm, the two operation images 1 and 2, which are of equal size, need to be provided. 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 operation images 1 and 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 (e.g. has only entries being either 0 for Black or 255 for White), the resultant 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 $P \subseteq \mathcal{N} \times \mathcal{N}$:

$$\begin{aligned} g_1 : \quad & P \to \{0, \ldots, g_{max}\} \\ g_2 : \quad & P \to \{0, \ldots, g_{max}\} \end{aligned} \tag{1}$$

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 $\{0, \ldots, g_{max}\} \times \{0, \ldots, g_{max}\}$. Then, the resultant image function is given by:

$$\begin{aligned} r \quad &: \quad P \to S \\ r(x, y) \quad &= \quad l(g_1(x, y), g_2(x, y)). \end{aligned} \tag{2}$$

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. Since the goal image is supplied as a binary one and in order to keep user instruction as simple as possible, in the following the 2D-Lookup matrix contains only binary entries Black (0) and White (255).

2.3 2D-Histogram-based 2D-Lookup

The 2D-Histogramm lookup is the 2D-Lookup algorithm with using the normalized 2D-Histogram as lookup matrix. The 2D-Histogram is defined as a matrix $H(g_1, g_2)$, derived from two operation images I_1 and I_2 of same size and resolution. An entry counts the number of occurrences of the same grayvalue pair at the same position (x, y) in both images. Thus

$$H(g_1, g_2) = \sum_{(x,y) \in I_1 \cap I_2} \delta(I_1(x, y), g_1)\delta(I_2(x, y), g_2) \tag{3}$$

with $\delta(a, b)$ being 1 for $a = b$ and 0 otherwise.

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 itself 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 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.

2.4 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 operations out of the set Squaring, Square Root, Move, Ordered Weighted Averaging (OWA [7]), fuzzy T-Norm, and Fuzzy Integral [6] for nodes of arity one, and pixel-wise addition, subtraction, multiplication and T-Norm for nodes of arity two (higher nodes are not used). The parameters of such a node operation are given by a 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 [1] for more details on the operation specifications.

In the framework presented here, the operator selection is performed four times, two for the extension to get the operation images, from which the 2D-Lookup matrix is derived, and two operations to get the operation images, for which the 2D-Lookup matrix is adapted for the goal image. In all four cases, the operation trees and the parameter structures are randomly initialized, and adapted later on by Genetic Programming [2] [3] as optimization procedure to gain high fitness values.

2.5 Operator fitness

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. For such a comparison, two sets are given, the reference set of the goal image and the pattern set of the result image.

In the following, $countBlack$ 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:

$$r_1 = \frac{matchBlack}{refBlack} \tag{4}$$

is the amount of reference pixels matched by the pattern,

$$r_2 = 1.0 - \frac{countBlack - matchBlack}{N - refBlack} \tag{5}$$

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

$$r_3 = \frac{matchBlack}{countBlack} \tag{6}$$

is the amount of matching pixels of the result image. 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:

$$f = 0.1r_1 + 0.5r_2 + 0.4r_3 \tag{7}$$

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, since r_2 and r_3 are 1 in this case; (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. In 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.

In the following, it is assumed, that the two operation images and the goal image are given. The question is how to derive a suitable 2D-Lookup matrix, which gives the best match between goal image and result image by the fitness measure given in the last subsection. The interesting point here is, that this derivation can be done be reusing this fitness measure. To prove this, assume a 2D-Lookup matrix, where all but one positions are set to White (1), and only a single position (g_1, g_2) is set to Black (0). Then, the 2D-Lookup will give a resultant image with all positions (x, y) set to Black, for which operation 1 yielded pixel value g_1 and operation 2 yielded pixel value g_2. Usually, there will be only a few black pixels within the result image. Now, the fitness measure will give values above 0.9, if the set of black pixels lies completely within the reference, no matter, how many pixels are there. So, a criterion can be given for setting a pixel to Black or White in the 2D-Lookup matrix.

Let $l_{(x,y)}$ be a 2D-Lookup matrix constituted by setting only the pixel at (x, y) to Black, and $r_{(x,y)}$ be the result of the 2D-Lookup with the operation images 1 and 2 and this 2D-Lookup matrix. Then

$$l(x,y) = \begin{cases} \text{Black}, & \text{if } f(r_{(x,y)}) > 0.88 \\ \text{White} & \text{otherwise.} \end{cases}$$

In case there are no black pixels in $r_{(x,y)}$ at all, $l(x, y)$ is set to Gray, which stands for positions within the 2D-Lookup matrix, whose pixel value pairs do never occur within the operation images 1 and 2 at the same location. The value 0.88 has been chosen instead of 0.9 to give the adaptation some tolerance.

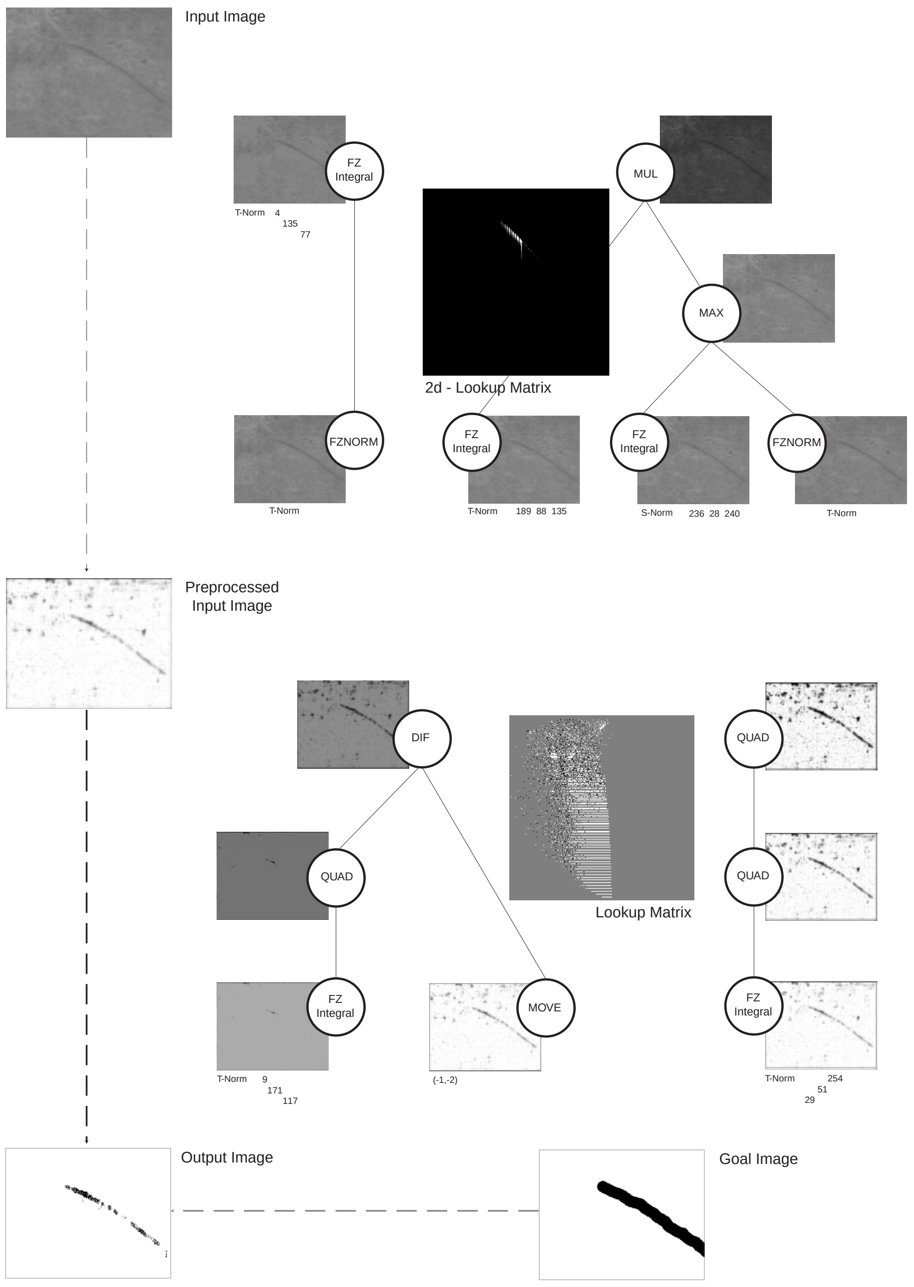

Figure 3: Complete example for the extended framework adapted to the low-contrast stroke in fig. 1.

3 Examples

The framework for 2D-Lookup adaptation was extended in order to enable the processing of low-contrast foregrounds. Figure 3 gives a complete example for such an adapted filtering. This example demonstrates several things:

- The final result image in the lower left is much more similar to the goal image than in fig. 1(c).
- 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.
- The operation trees are not ”smart” in the sense that parts of it 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 fig. 3) basically produces a gray image, from which a shifted version of the preprocessed image is subtracted. Such parts can be removed in a re-design phase of the filters after adaptation by hand (especially when they involve computationally more expensive node operations like the Fuzzy Integral).
- The 2D-Lookup for this example is basically a lookup with the co-occurrence matrix of the preprocessed image.
- 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 4 shows another examples, 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.

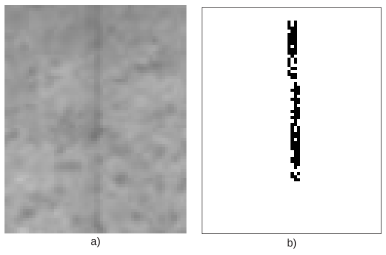

Figure 4: Result of the extended framework for vibrating-knife fault.

Another application field for the extended framework could be handwriting separation on textured background. Figure 5 shows an example for this. However, obviously the performance here is worse than for the framework without extension (subfigure (b) without preprocessing, (c) with preprocessing). The reason lies in the preprocessed image (d), which is not related to the goal image at all: parts of handwriting have been removed, since their local appearance is typical for the whole image, especially when handwriting stroke direction fits the direction of the lines in the background pattern. Thus, the consecuting 2D-Lookup adaptation is not able to recover those parts from the preprocessed image.

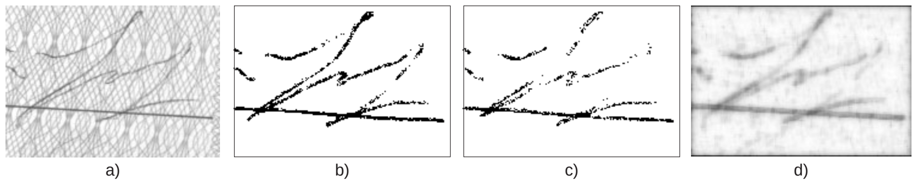

Figure 5: Extended framework applied to handwriting separation shows lower performance.

4 Summary

This paper presented a framework for low-contrast texture fault processing, i.e. faults which are hard to separate from the background texture. The framework is a two-stage one, based on 2D-Histogram lookup and consecuting 2D-Lookup adaptation. 2D-Histogram lookup itself is a variant of 2D-Lookup operation. For 2D-Lookup, in two differently processed versions of the same image, grayvalue pairs from the same location in both images are replaced (”looked-up”) by the corresponding entry in a given two-dimensional matrix. The two operations and the matrix have to be provided for full algorithm specification. This gives the algorithm the 2D-Lookup algorithm high versatility for the application in various contexts. For 2D-Histogram lookup, the matrix is derived from the 2D-Histogram of both processed images. The main advantage of using the 2D-Histogram is the darkening of rarely occurring structures in the image, while highly probable image structures becomes bright. The so-processed images are then given to a 2D-Lookup procedure for automatic filter generation, employing Genetic Programming for the generation of appropriate image processing operations. For low-contrast texture faults, the approach shows better performance in fault region detection than the approach of 2D-Lookup adaptation without 2D-Histogram lookup. Performance in case of handwriting was not so good due to common appearance of patterns in both, the handwriting and the background.

References

[1] Mario Köppen and Bertram Nickolay. Genetic programming based texture filtering framework. In Nikhil R. Pal, editor, *Pattern Recognition in Soft Computing Paradigm*, FLSI Soft Computing Series - Vol. 2, pages 275–304. World Scientific, 2001.

[2] John R. Koza. *Genetic Programming – On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, London MA, 1992.

[3] John R. Koza. *Genetic Programming II – Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, London MA, 1994.

[4] Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.

[5] Jean Serra. *Image Analysis and Mathematical Morphology 2: Theoretical Advances*. Academic Press, London, 1988.

[6] M. Sugeno. *Fuzzy Control*. Nikkan Kogyo Shimbun-sha, 1988. (in Japanisch).

[7] R.R. Yager. On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transaction on System, Man & Cybernetics*, 18:183–190, 1988.