PIXEL CLASSIFICATION: A FUZZY-GENETIC APPROACH

Ulrich Bodenhofer, Erich Peter Klement

Fuzzy Logic Laboratorium Linz-Hagenberg
Institut für Mathematik
Johannes Kepler Universität, A-4040 Linz, Austria
ip@flll.uni-linz.ac.at

Abstract. In this paper a fuzzy method for pixel classification is proposed. It is one of the most important results of the development of an inspection system for a silk-screen printing process. The classification algorithm is applied to a reference image in the initial step of the printing process in order to obtain regions which are to be checked by applying different criteria. Tight limitations in terms of computation speed have necessitated very specific, efficient methods which operate locally. These methods are motivated and presented in detail in the following. Furthermore, the optimization of the parameters of the classification system with genetic algorithms is discussed. Finally, the genetic approach is compared with other probilistic optimization methods.

Keywords: discrepancy norm, edge detection, genetic algorithm, pixel classification

1 Introduction

The main goal of this project was to design an automatic inspection system which does not sort out every print with defects, but only those which have visible defects in a way which is really unacceptable for the consumer. It is clear that the visibility of a defect depends on the structure of the print in its neighborhood. While little spots can hardly be recognized in very chaotic areas, they can be disturbing in rather homogeneous areas. So, the first step towards a sensitive inspection is to partition the print into areas of different sensitivity which, consequently, should be treated differently.

The following types were specified by experts of our partner company. For certain reasons, which can be explained with the special principles of the silk-screen printing process, it is sufficient to consider only these four types:

Homogeneous area: uniformly colored area;
Edge area: pixels within or close to visually significant edges;
Halftone: area which looks rather homogeneous from a certain distance, although it is actually obtained by printing small raster dots of two or more colors;
Picture: rastered area with high, chaotic deviations, in particular small high-contrasted details.

The magnifications in Figure 1 show how these areas typically look like at the pixel level. Of course, transitions between two or more of these areas are possible, hence a fuzzy model is recommendable.

First of all, we should define precisely what, in our case, an image is:

Definition 1 *An $N \times M$ matrix of the form*

$$((u_r(i,j), u_g(i,j), u_b(i,j)))_{i=1,\ldots,N}^{j=1,\ldots,M} \qquad (1)$$

with three-dimensional entries (additive RGB model)

$$(u_r(i,j), u_g(i,j), u_b(i,j)) \in \{0,\ldots,255\}^3$$

is a model of a 24 bit color image of size $N \times M$. A coordinate pair (i,j) stands for a pixel; the values $(u_r(i,j), u_g(i,j), u_b(i,j))$ are called the gray values of pixel (i,j).

It is near at hand to use something like the variance or an other measure for deviations to distinguish between areas which show only low deviations, such as homogeneous areas and halftone areas, and areas with rather high deviations, such as edges or pictures.

On the contrary, it is intuitively clear that such a measure can never be used to separate edge areas from picture areas, because of the neglection of any geometrical information.

Experiments have shown that well-known standard edge detectors, such as the Laplacian or the Mexican Hat, but also many other locally operating filter masks (see e.g. [7]), cannot distinguish sufficiently if deviations are chaotic or anisotropic.

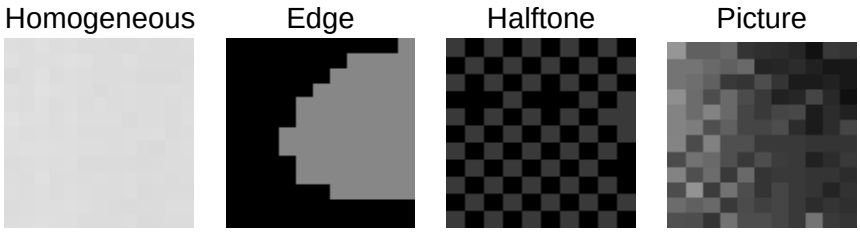
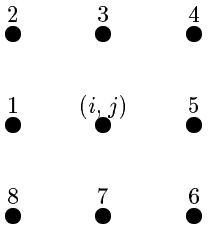


Figure 1: Magnifications of typical representatives of the four types

Another possibility we also took into consideration was to use wavelet transforms or more sophisticated image segmentation methods (see for instance [7] or [3]). Since we had to cope with serious restrictions in terms of computation speed, such highly advanced methods, although they are efficient, would require too much time. Finally, we found a fairly good alternative which is based on the discrepancy norm. This approach uses only, like the simpliest filter masks, the closest neighborhood of a pixel.

For an arbitrary but fixed pixel (i, j) we defined the enumeration mapping l as follows:



k	$l(k)$
1	$(\quad i \quad, j-1)$
2	$(i-1, j-1)$
3	$(i-1, \quad j \quad)$
4	$(i-1, j+1)$
5	$(\quad i \quad, j+1)$
6	$(i+1, j+1)$
7	$(i+1, \quad j \quad)$
8	$(i+1, j-1)$

If we plot one color extraction of the eight neighbor pixels with respect to this enumeration, i.e. $(u_x(l(k)))_{k\in\{1,\ldots,8\}}$, where $x \in \{r, g, b\}$, we typically get curves like those shown in Figure 2.

From these sketches it can be seen easily that a measure for the deviations can be used to distinguish between homogeneous areas, halftones, and the other two types. On the other hand, the most eyecatching difference between the curves around pixels in pictures and edge areas is that, in the case of an edge pixel, the peaks appear to be more connected while they are mainly chaotic and narrow for a pixel in a picture area. So, a method which judges the shape of the peaks should be used in order to separate edge areas from pictures. A simple but effective method for this purpose is the so-called discrepancy norm, for which there are already other applications in pattern recognition (cf. [6]).

2 The Discrepancy Norm

Definition 2 *The mapping*

$$\begin{aligned} \|.\|_D : \quad \mathbb{R}^\kappa &\longrightarrow \mathbb{R}^+ \\ \vec{x} &\longmapsto \max_{1\le\alpha\le\beta\le n} \left| \sum_{i=\alpha}^{\beta} x_i \right| \end{aligned} \tag{2}$$

is called discrepancy norm on $\mathbb{R}^\kappa$.

In words, $\|\vec{x}\|_D$ is the absolute value of the maximal sum of consecutive coordinates of the vector $\vec{x}$. Obviously, unlike conventional norms, the signs and the order of the entries play an essential role. Nevertheless, the mapping $\|.\|_D$ is a norm on $\mathbb{R}^\kappa$.

It can be seen easily that the more entries with equal signs appear successively, the higher the discrepancy norm is. On the contrary, for sequences with alternating signs it is close to the supremum norm. Therefore, $\|.\|_D$ can be used for judging the connectedness of the peaks with equal signs.

Obviously, the computation of $\|.\|_D$ by strictly using the definition requires $\mathcal{O}(n^2)$ operations. The following theorem allows us to compute $\|.\|_D$ with linear speed:

Theorem 1 *For all* $\vec{x} \in \mathbb{R}^\kappa$ *we have*

$$\|\vec{x}\|_D = \max_{1\le\beta\le n} X_\beta - \min_{1\le\alpha\le n} X_\alpha, \tag{3}$$

where the values $X_j = \sum_{i=1}^{j} x_i$ *denote the partial sums.*

A more detailed analysis of the discrepancy norm can be found in [2].

3 The Fuzzy System

For each pixel (i, j) we consider its nearest eight neighbors enumerated as described above, which yields three vectors of gray values with 8 entries — one for each color extraction. As already mentioned, the sum of the variances of the three vectors

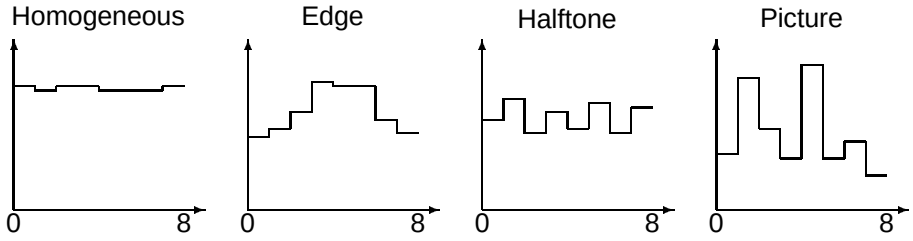
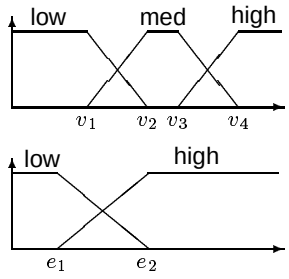


Figure 2: Typical gray value curves of the form $(u_x(l(k)))_{k\in\{1,\ldots,8\}}$

can be taken as a measure for the size of the deviations in the neighborhood of the pixel. Let us denote this value with $v(i,j)$. On the other hand, the sum of the discrepancy norms of the vectors, where we subtract each entry by the mean value of all entries, can be used as a criterion whether the pixel is within or close to a visually significant edge.

Of course, e itself can be used as an edge detector. Figure 3 shows how good it works compared with the commonly used Mexican Hat filter mask.

The fuzzy decision is then carried out for each pixel (i,j) independently: First of all, the characteristic values $v(i,j)$ and $e(i,j)$ are computed. These values are taken as the input of a small fuzzy system with two inputs and one output. Let us denote the linguistic variables on the input side with v and e. Since the position of the pixel is of no relevance for the decision in this concrete application, indices can be omitted here. The input space of the variable v is represented by three fuzzy sets which are labeled "low", "med", and "high". Analogously, the input space of the variable e is represented by two fuzzy sets, which are labeled "low" and "high". Experiments have shown that $[0,600]$ and $[0,200]$ are appropriate universes of discourse for v and e, respectively. For the decomposition of the input domains simple Ruspini partitions (see [8]) consisting of trapezoidal fuzzy subsets were chosen:



The output space is a set of linguistic labels, namely "Ho", "Ed", "Ha", and "Pi", which are, of course, just abbreviations of the names of the four types. Let us denote the output variable itself with t. Finally, the output of the system for each pixel (i,j) is a fuzzy subset of {"Ho", "Ed", "Ha", "Pi"}. This output set is computed by processing the values $v(i,j)$ and $e(i,j)$ through a rulebase with five rules, which cover all the possible combinations:

IF	v is low			THEN	$t =$ Ho
IF	v is med	AND	e is high	THEN	$t =$ Ed
IF	v is high	AND	e is high	THEN	$t =$ Ed
IF	v is med	AND	e is low	THEN	$t =$ Ha
IF	v is high	AND	e is low	THEN	$t =$ Pi

In this application ordinary Mamdani min/max-inference is used. Finally, the degree to which "Ho", "Ed", "Ha", or "Pi" belong to the output set can be regarded as the degree to which the particular pixel belongs to area Homogeneous, Edge, Halftone, or Picture, respectively.

4 Experimental Results

In our concrete application we consider a clipping of approximately 200000 pixels. For such an image the classification takes at most three seconds on the hardware which had to be used (a standard workstation with a 150MHz RISC CPU). For the given requirements the classifications are satisfactory in almost all cases which appear in every day printing.

The classification is done in the setup phase of the printing process, which may take at most four seconds. As already mentioned, the proposed methods are especially suited for the needs of this specific application. The price to be paid for the applicability of the methods under such heavy time constraints is a certain loss of universality.

For more details on the integration of the classification algorithm into the printing process see [1].

5 The Optimization of the Classification System

The behavior of the fuzzy system depends on six parameters, $v_1,\ldots,v_4$, e_1, and e_2, which deter-

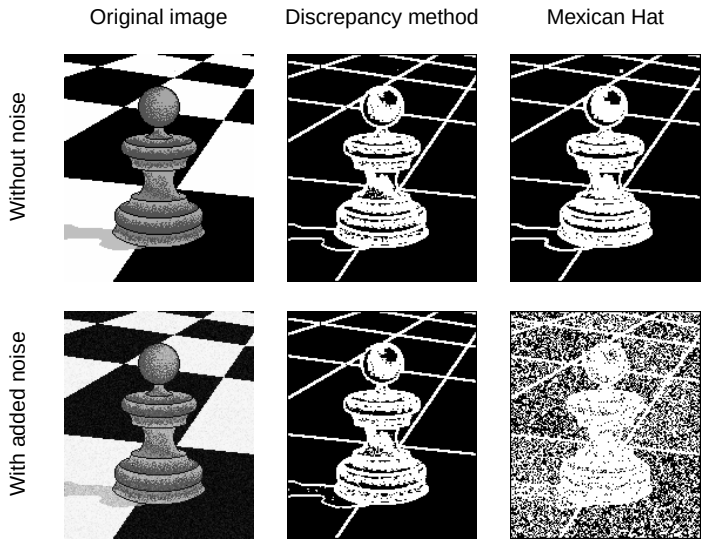


Figure 3: Comparison between e and a standard 3×3 filter mask

mine the shape of the two fuzzy partitions. In the first step these parameters were tuned manually. Of course, we have also taken into consideration the use of (semi)automatic methods for finding the optimal parameters.

The general problem is not to find an appropriate algorithm for doing that task, the difficulty is how to judge such a classification. Since the specification of the four types of areas is given in a vague, verbal form, no mathematical criterion is available for that. Hence, a model-based optimization process is, because of the lack of a model, not applicable. The alternative is a knowledge-based approach, which poses the question how to generate this knowledge — the examples from which the algorithm should learn.

Our optimization procedure consists of a painting program which offers tools, such as a pencil, a rubber, a filling algorithm, and many more, which can be used to make a classification of a given representative image by hand. Then an optimization algorithm can be used to find that configuration of parameters which yields the maximal degree of matching between the desired result and the output actually obtained by the classification system.

Assume that we have N sample pixels for which the pairs of input values $(\tilde{v}_k, \tilde{e}_k)_{k\in\{1,..,N\}}$ are computed and that we already have a reference classification of these pixels $\tilde{t}(k) = (\tilde{t}_{\mathsf{Ho}}(k), \tilde{t}_{\mathsf{Ed}}(k), \tilde{t}_{\mathsf{Ha}}(k), \tilde{t}_{\mathsf{Pi}}(k))$, where $k \in \{1, \ldots, N\}$. Since, as soon as the values $\tilde{v}$ and $\tilde{e}$ are computed, the geometry of the image plays no role anymore, we can switch to one-dimensional indices here. Then one possibility to define the performance (fitness) of the fuzzy system would be

$$\frac{1}{N}\sum_{k=1}^{N} d(t(k), \tilde{t}(k)), \tag{4}$$

where $t(k) = (t_{\mathsf{Ho}}(k), t_{\mathsf{Ed}}(k), t_{\mathsf{Ha}}(k), t_{\mathsf{Pi}}(k))$ are the classifications actually obtained by the fuzzy system for the input pairs $(\tilde{v}_k, \tilde{e}_k)$ with respect to the parameters v_1, v_2, v_3, v_4, e_1, and e_2; $d(.,.)$ is an arbitrary (pseudo-)metric on $[0,1]^4$. The problem of this brute force approach is that the output of the fuzzy system has to be evaluated for each pair (v_k, e_k), even if many of these values are similar or even equal. In order to keep the amount of computation low, we "simplified" the procedure by a "clustering process" as follows:

We choose a partition $(P_1, \ldots, P_K)$ of the input space and count the number $(n_1, \ldots, n_K)$ of sample points $\{p_1^i, \ldots, p_{n_i}^i\}$ each part contains. Then the desired classification of a certain part (cluster) can be defined as

$$\tilde{t}_X(P_i) = \frac{1}{n_i}\sum_{j=1}^{n_i} \tilde{t}_X(p_j^i), \tag{5}$$

where $X \in \{\mathsf{Ho}, \mathsf{Ed}, \mathsf{Ha}, \mathsf{Pi}\}$.

If ϕ is a function which maps each cluster to a representative value (e.g., its center of gravity), we can define the fitness (objective) function as

$$\frac{100}{N}\sum_{i=1}^{K} n_i \cdot \left(1 - \frac{1}{2}\cdot d_i\right), \tag{6}$$

where

$$d_i = \sum_{X \in \{\text{Ho, Ed, Ha, Pi}\}} \left(\tilde{t}_X(P_i) - t_X(\phi(P_i))\right)^2.$$

If the number of parts is chosen moderately (e.g. a rectangular 64×32 net which yields $K = 2048$) the evaluation of the fitness function takes considerably less time than a direct application of formula (4).

Note that in (6) the fitness is already transformed such that it can be regarded as a degree of matching between the desired and the actually obtained classification measured in percent. This value has then to be maximized.

In fact, fitness functions of this type are, in almost all cases, continuous but not differentiable and have a lot of local maxima. Therefore, it is more reasonable rather to use probabilistic optimization algorithms than to apply continuous optimization methods, which make excessive use of derivatives. This, first of all, requires a (binary) coding of the parameters. We decided to use a coding which maps the parameters v_1, v_2, v_3, v_4, e_1, and e_2 to a string of six 8-bit integers $s_1, \ldots, s_6$ which range from 0 to 255. The following table shows how the encoding and decoding is done:

$$\begin{array}{lllllll} s_1 & = & v_1 & & v_1 & = & s_1 \\ s_2 & = & v_2 - v_1 & & v_2 & = & s_1 + s_2 \\ s_3 & = & v_3 - v_2 & & v_3 & = & s_1 + s_2 + s_3 \\ s_4 & = & v_4 - v_3 & & v_4 & = & s_1 + s_2 + s_3 + s_4 \\ s_5 & = & e_1 & & e_1 & = & s_5 \\ s_6 & = & e_2 - e_1 & & e_2 & = & s_5 + s_6 \end{array}$$

A class of probabilistic optimization methods which has come into fashion in the last years are genetic algorithms (GAs). They can be regarded as simplified simulations of an evolution process, based on the principles of genetic reproduction employing mechanisms, such as selection, mutation, and sexual reproduction. Another important difference between GAs and conventional optimization algorithms is that GAs do not operate on single points but on whole populations of points (which, in this case, are binary strings).

We first tried a standard GA (see [4] or [5]) with proportional (standard roulette wheel) selection, one-point crossing over with uniform selection of the crossing point, and bitwise mutation. The size of the population m was constant, the length of the strings was 48 (compare with the coding above, see [4] for an overview of more sophisticated variants of GAs).

In order to compare the performance of the GAs with other well-known probabilistic optimization methods, we additionally considered the following methods:

Hill climbing: always moves to the best-fitted neighbor of the current string until a local maximum is reached; the initial string is generated randomly.

Simulated annealing: powerful, often used probabilistic method which is based on the imitation of the solidification of a crystal under slowly decreasing temperature (see [9] for a detailed description)

Each one of these methods requires only a few binary operations in each step. Most of the time is consumed by the evaluation of the fitness function. So, it is near at hand to take the number of evaluations as a measure for the speed of the algorithms.

Results

All these algorithms are probabilistic methods, therefore their results are not well-determined, they can differ randomly within certain boundaries. In order to get more information about their average behavior, we tried out each one of them 20 times for one certain problem. For the given problem we found out that the maximal degree of matching between the reference classification and the classification actually obtained by the fuzzy system was 94.3776% . See the table in Figure 4. In this table $f_{\max}$ is the fitness of the best and $f_{\min}$ is the fitness of the worst solution; $\bar{f}$ denotes the average fitness of the 20 solutions, σ_f denotes the standard deviation of the fitness values of the 20 solutions, and # stands for the average number of evaluations of the fitness function which was necessary to reach the solution.

The hill climbing method with a random selection of the initial string converged rather quickly. Unfortunately, it was always trapped in a local maximum, but never reached the global solution (at least in these 20 trials).

The simulated annealing algorithm showed similar behavior at the very beginning. After tuning the parameters involved, the performance improved remarkably.

The raw genetic algorithm was implemented with a population size of 20; the crossing over

	$f_{\max}$	$f_{\min}$	$\bar{f}$	σ_f	#
Hill Climbing	94.3659	89.6629	93.5536	1.106	862
Simulated Annealing	94.3648	89.6625	93.5639	1.390	1510
Improved Simulated Annealing	94.3773	93.7056	94.2697	0.229	21968
GA	94.3760	93.5927	94.2485	0.218	9910
Hybrid GA (elite)	94.3760	93.6299	94.2775	0.207	7460
Hybrid GA (random)	94.3776	94.3362	94.3693	0.009	18631

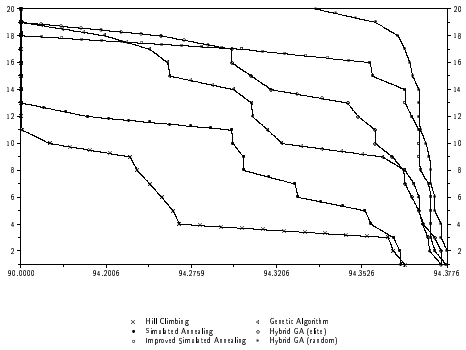


Figure 4: A comparison of the results of various probabilistic optimization methods

probability was set to 0.15, the mutation probability was 0.005 for each byte. It behaved pretty well from the beginning, but it seemed inferior to the improved simulated annealing.

Next, we tried a hybrid GA, where we kept the genetic operations and parameters of the raw GA, but every 50th generation the best-fitted individual was taken as initial string for a hill climbing method. Although the performance increased slightly, the hybrid method still seemed to be worse than the improved simulated annealing algorithm. The reason that the effects of this modification were not so dramatic might be that the probability is rather high that the best individual is already a local maximum. So we modified the procedure again. This time a *randomly chosen individual* of every 25th generation was used as initial solution of the hill climbing method. The result exceeded the expectations by far. The algorithm was, in all cases, nearer to the global solution than the improved simulated annealing was (compare with table in Figure 4) but, surprisingly, sufficed with less invocations of the fitness function. The graph in Figure 4 shows the results graphically. Each line in this graph corresponds to one algorithm. The curve shows, for a given fitness value x, how many of the 20 different solutions had a fitness higher or equal to x. It can be seen easily from this graph that the hybrid GA with random selection led to the best results. Note that the x-axis is not a linear scale in this figure. It was transformed in order to make small differences visible.

6 Conclusion

In the first part of this paper we demonstrated the synergy which lies in the combination of fuzzy systems with, more or less, conventional methods. This combination is in particular suitable for designing specific algorithms for time-critical problems. However, this specifity often results in a loss of universality.

In the second part we showed the suitability of genetic algorithms for finding the optimal parameters of a fuzzy system, especially if the analytical properties of the objective function are bad. Moreover, hybridization has been discovered as an enormous potential for improvements of genetic algorithms.

References

1. P. Bauer, U. Bodenhofer, E. P. Klement. “A fuzzy method for pixel classification and its application to print inspection”. Proc. IPMU'96, Granada, July 1996, pp. 1301–1305.
2. P. Bauer, U. Bodenhofer, E. P. Klement. “A fuzzy algorithm for pixel classification based on the discrepancy norm”. Proc. FUZZ-IEEE'97, New Orleans, September 1996, pp. 2007–2012.
3. J. C. Bezdek, S. K. Pal. *Fuzzy Models for Pattern Recognition*. IEEE Press, New York, 1992.
4. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.
5. J. H. Holland. *Adaptation in Natural and Artificial Systems*. The MIT Press, Cambridge, MA, 1992.
6. H. Neunzert, B. Wetton. “Pattern recognition using measure space metrics”. Technical Report 28, Universität Kaiserslautern, Fachbereich Mathematik, November 1987.
7. A. Rosenfeld, A. C. Kak. *Digital Picture Processing*, volume II. Academic Press, San Diego, CA, second edition, 1982.
8. E. H. Ruspini. “A new approach to clustering”. *Information & Control*. **15**, 22–32, 1969.
9. P. J. M. van Laarhoven, E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers, Dordrecht, 1987.