Lightweight Evolutionary Computation beating the Complexity of Image-Processing Applications

Mario Köppen, Katrin Franke and Raul Vicente-Garcia
Fraunhofer IPK Berlin, Pascalstr. 8-9, 10587 Berlin, Germany
$mario.koeppen, franke, raul@ipk.fraunhofer.de$

Abstract

The expedience of today's image-processing applications is not any longer based on the performance of a single algorithm alone. These systems appear to be complex frameworks with a lot of subtasks that are solved by specific algorithms, adaptation procedures, data handling, scheduling, and parameter choices. The venture of using computational intelligence (CI) in such a context, thus, is not a matter of a single approach. Among the great choice of techniques to inject CI in an image-processing framework, the primary focus of this presentation will be on the usage of so-called Tiny-GAs. This stands for an evolutionary procedure with low efforts, i.e. small population size (like 10 individuals), little number of generations, and a simple fitness. Obviously, this is not suitable for solving highly complex optimization tasks, but the primary interest here is not the best individuals' fitness, but the fortune of the algorithm and its population, which has just escaped the Monte-Carlo domain after random initialization. That this approach can work in practice will be demonstrated by means of selected image-processing applications, especially in the context of linear regression and line fitting; evolutionary post processing of various clustering results, in order to select a most suitable one by similarity; and classification by the fitness values obtained after a few generations.

1 Introduction

In order to recognize a visual object by means of the computer, many processing steps are needed. These steps barely resemble the nature of human visual perception, but are rather crude artificial replacements. And although much research and development was conducted over the past decades, computer vision is still a great challenge. In general, the common scheme for visual object recognition comprises image acquisition, preprocessing, feature extraction and selection, learning and classification. Figure 1 displays such an image-processing system and the means for its development. The typical goal here is the transition from raw data captured from a real world scene by means of an image-acquisition device, e.g. CCD camera, CMOS camera, scanner, other sensors, into the abstract linguistic world of object names, relation between objects, meanings, content and interpretation – mimicking the human ability of handling visual stimuli within a purely technical domain. The intermediary steps of converting raw data into more abstract object features, and features' consecuting assignment to "class bins" offer a great deal of freedom, uncertainty and variation. Nothing in the object-recognition framework is uniquely specified, nor is there any procedure known to derive the way that features are computed or classified from the application itself. With regard to this highly multi-modal context and the increase of complexity in modern image-processing application, automated selection and optimization procedures are needed obviously.

Its not a surprise that the usage of Evolutionary Computation (EC) in this field has gained a lot of attraction in the past two decades. EC makes it possible to design solutions for optimization problems. According to EC's natural role model, life forms are enabled to adapt to particular environmental changes over successive generations. The mechanisms that drive natural evolution are reproduction, mutation and survival of the fittest. From a computational point of view this can be seen as an optimization process. The application of evolution mechanisms by artificial / computational systems is called EC. Therefore, one can hold that "EC takes the power of natural selection to turn computers into automated optimization tools." EC algorithms are efficient, adaptive and robust search procedures, producing nearly optimal solutions, and they show a high degree of implicit parallelism. General approaches and methods comprise Genetic Algorithms [1, 2, 3], Genetic Programming [4, 5] and Evolutionary Strategy [6]. Many applications of Computational Intelligence in image processing and pattern recognition have been presented in the past two decades. The impossibility to provide a complete survey here (a technical report of the University of Vaasa from 1999 [7] already lists over 1000 publications in this field) forces us to pick up just a few representative fields, as e.g. in image filtering, incl. noise removal [8], design of wavelet filters [9], morphological filters [10, 11], or general design of image operators [12, 13]; image compression [14, 15, 16]; in biometrics: pre-processing of fingerprints [17], or within the more general theme of "soft biometrics" [18, 19]; clustering and segmentation [20, 21, 22, 23]; object detection and recognition [24, 25]; feature extraction [26, 27]; classifier fusion [28, 29, 30]; 3D-image processing, esp. registration of 3D objects [31], matching [32], recognition [33], or stereo matching [34, 35]; edge detection [36, 37]; camera calibration [38, 39]; image synthesis [40], visual routines [41], digital watermarking of still images [42, 43]; optimization of super-resolution [44]; and detection of faults in video image transmission [45].

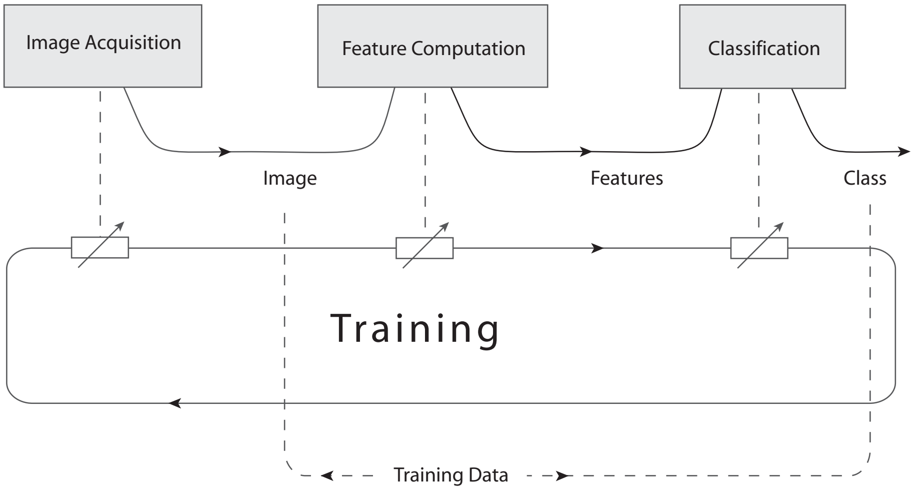


Figure 1: Development of an image processing system.

The higher the complexity of the problem domain, or the less one knows about ways to model it, the more affordable it seems using EC. In this sense, EC can be also applied to the framework of fig. 1: it can be used to select and adapt the feature computation, to train the classifier, or to support the preprocessing of the just-acquired image to yield better object segmentation and consequently better separability of the features with respect to the classes. In such cases, the optimization goal is derived from an application-driven input-output behaviour that involves mass-data processing. Typically, the amount of image-raw data is in the range of a million bytes of information. However, this does not automatically mean that the EC algorithms have to be large-scaled as well. The opportunities that EC algorithms can offer on a smaller scale are explored in our studies. We investigate *lightweight* evolutionary algorithms that are characterized by the following features:

- small population size,
- little number of generations, and
- simple fitness.

On first glance, this concept seems futile. What can be expected at all from such *lightweight* algorithms (lets call them Tiny-GAs in the following) in the highly complex domain of image processing? We expose some pros and contras as follows:

- Such algorithms can be used online: Often, image-processing frameworks have two operation modes, online and offline. The offline operation mode includes the interactive process of configuring the system, its *training*, to championize it for the later online processing mode, where there is usually hardly possible to modify any of its internal settings, and where there are often strong processing-time restrictions. The high effort of training is reflected by the opportunity to use high-effort training procedures, making it impossible to use such algorithms in the online mode in the same manner. However, compared to the compuation time needed for the bulk processing of raw image data, smaller ECs may become competitive with other image-processing algorithms, from which the application framework is composed.
- They will not solve complex partial problems, and the evolved values of the optimization function will be of no value for the framework configuration. This is obviously a strong statement against the use of such algorithms, but neglects the fact that the evolving population can provide other information as well. As in a formula-1-car race, where most of the audience is watching the fortune of the top three, four leading cars only, equipped with world-top drivers: this oversees the fact that there might be much more interesting going on in the middle field of the race, where experienced drivers push the best out of their weaker cars, or less good drivers are handling good cars with low fortune. In a similar sense, the "middle field" of an ECs population is sometimes worth a closer look as well.
- They may simplify the framework: a number of sensitive parameter settings that are necessary to ensure reliable operation of the system in online mode can be integrated into a well-designed fitness function, thus making the "external" setting of parameters in offline mode before dispensable.
- The primary focus of the heuristic has to be given on exploration and verification, and not on exploitation. Typically, EC algorithms are randomly initialized, with letting some *a priori*

knowledge of the optimization task giving a means for influencing the distribution of these random values. It can be also said that they are grounded in pure Monte-Carlo terrain, and after a few generations, we might find them just airborne... and as for a starting plane, from which we have just borrowed the analogy, this changes a lot regarding stability, speed, operation and maintenance, but will not change much more during further flight operation. So, some characteristics of an EC algorithm might not change much more with the increasing number of generations, once the population has already entered a promising region of the search space. As will be seen below, this is mostly an issue of making it probable to find such a region, and refraining from any "needle in a haystack"-like problem specifications.

- They may become dispensable during the further development of the framework. This is standing for the fact that other, pure non-EC image-processing algorithms may become better specified from gaining more domain knowledge, and be it from the initial use of a simple EC itself - but this may qualify our Tiny-GAs just as a suitable prototyping technique, and nothing more.
- Such algorithms, in conjunction with the specific application domain and their lower dimensions, can probably be better understood and analyzed, and their runtime behaviour can be better predicted.

In the following, we are going to study the concept of a Tiny-GA in the context of image-processing applications in more detail. First, we pick-up a random example, line-fitting to data points, to justify the potential. It will guide to the need of specifying a typical class of combinatorial optimization problems appearing in such an approach, and that does not seem to be well-studied so far. Roughly spoken, Tiny-GAs can do a good job in *subset-selection* with complex selection criterias. The mathematical notion here is the *Generalized Bitstring Prototype Problem* that we can identify in several typical image-processing-problem classes like regression, classification, clustering and segmentation. At the end, its the proper organization and structure of the used framework itself that accounts the affordability of smaller ECs. Authors hope that this study is just seen as a first step, and that it will stimulate further research and developments also going into this direction.

2 Linear Regression

During the development of a real-world image processing application, there are a number of recurring problems. Among them are regression problems: some pre-processing of the image has lead to the identification of a number of image positions that are, yet not perfectly but nearly, located on a model curve. The task is to find a symbolic expression for such a model curve.

In the small problem that we want to study here, the points were the result of the pre-processing of ID cards. In the bottom area of such documents, there is the so-called *machine-readable zone* (MRZ), where the information of the document is encoded and printed using a font that supports easy readability with standard optical character recognition methods. After manual scanning with resolution of 300dpi, the document might be slightly mis-oriented. We are about to tolerate orientation angles up to 30 degree, and thus have to acquire orientation information from the components of the scanned image before the processing can be continued. We will not go into much details of

Category	Description	Comment
Mathematical Morphology	closing with structuring element larger than distance between points; connected-component analysis; linear regression with points within the largest connected component only	processing time large; may fail if distance of co-linear points is sometimes large
Clustering	agglomerative clustering will link the sequence of nearly equidistant points belonging to one text row	same as for Mathematical Morphology: failures to detect some of the characters can produce larger gaps, and disturb the clustering
Hough Transform	default method to find lines connecting given points of images	needs a pre-setting of angle intervalls with a trade-off: the finer the granulation of the angle values, the more sparse the occupation of the accumulator cells
Discriminating Line	the plane is divided into two segments by a separating line thus that all nearly co-linear points are located in one half-plane	if this line is iteratively adapted, it needs a clever intialization; if not, several assumptions about the point distribution have to be made

Table 1: Some approaches to handle the partial regression problem shown in fig. 2.

the processing here, as we are just interested in the intermediate optimization problem that will arise. For the pre-processing, the image is binarized, and the connected components of the binary image are extracted. In the MRZ, these are the single character images. The points shown in fig. 2 then are the centers of the bounding boxes of each character component. As it can be seen, the majority of them is nearly co-linear, but due to the strong orientation, also some points of the second text row of the MRZ are extracted. Despite of these “outliers,” we are looking for a procedure to estimate the total document orientation angle from these points.

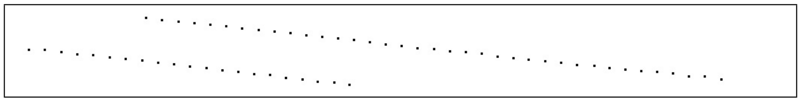


Figure 2: A simple problem? Goal is to estimate the rotation angle of the document from this set of points that has been extracted by pre-processing of an ID document image.

The problem looks easy, and several approaches will come into mind, with using an EC algorithm not being among the primary choices. Table 1 lists some of the possible approaches.

As can be seen, each approach will be able to handle such a problem. Their common drawback is that they need some *a priori* knowledge of the point alignment (expected offset of adjacent points, selection of rotation angle granulation; expected size of the gap between the two text rows represented by the points; initial guess of a point between these two rows), which usually results in several application parameter that have to be set carefully in advance to ensure smooth operation. Given that this is only one module of a larger number of modules comprising the whole processing framework, the number of parameters can become large. So, the objective of an alternative approach here could be to reduce the number of parameters while not increasing the computational effort, if compared to the given approaches.

We will consider now the use of an EC algorithm in this context. Therefore, the problem is stated as an optimization problem, a combinatorial one, to be more specific:

Given a set of points, select the largest subset with minimal average linear regression error.

More formally: Given a set of n points (x_i, y_i) in the plane $(n \geq 2)$, the task for linear regression is to find suitable parameters a and b such that the expression $y_i = ax_i + b$ fits the given data pairs as good as possible. The average regression error than can be computed as the mean-squared error of the mapping $y = f(x)$:

$$E(a,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - ax_i - b)^2 \tag{1}$$

The other objective is mandatory: if just two points are selected, there is a line connecting them with regression error zero. Once more than two points are selected, the well-known linear regression formula will provide the values for the slope a and bias b that minimise the average regression error.

That using the linear regression formula for the whole set of points is not sufficient for the case of a nearly co-linear subset of points with outliers, is illustrated in fig. 3.

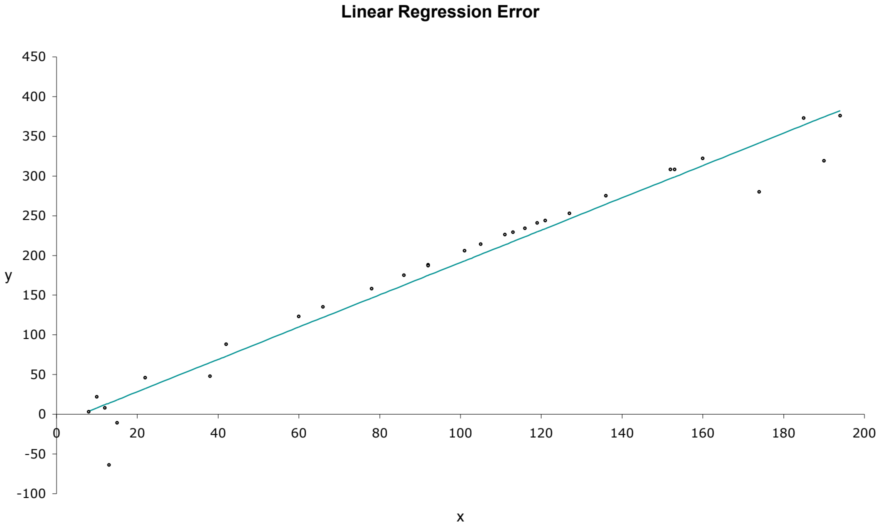


Figure 3: Problem with the Linear Regression: While most of the points are nearly co-linear, the regression line is remarkably shifted by the presence of a few outliers.

In summary, we have to express the optimization goal by *two objectives*, large subsets, and small average regression error. This motivates the use of an algorithm of the recently emerging field of Evolutionary Multi-Objective Optimization (EMO).

Without going into unnecessary details here, let us try to fuse the complex matter of multi-objective optimization into a few words. The main difference to the optimization for a single objective is the partial incomparability of results. If there are two objective functions $y_1 = f_1(x)$ and $y_2 = f_2(x)$, with the goal to find x that minimize both, y_1 and y_2, it is possible that we have, for two different choices of x, $x = x_a$ and $x = x_b$, the situation that $f_1(x_a) < f_1(x_b)$ but also $f_2(x_a) > f_2(x_b)$. So, for such x-values, a gain in one objective can only be achieved by loosing on the other. The set of all such points in the objective space (i.e. the space of the y-values) is usually called the PARETO front of the multi-objective optimization problem. The goal of an EMO algorithm is to sample the PARETO front as complete as possible.

One mandatory component of modern EMO algorithm is the *archive* [46]. During the generational steps of an EMO, the archive stores the PARETO set of the totality of points in objective space that have ever been visited by the algorithm. Initially, the archive is empty. Each time, a new individual is created in the algorithm (be it by mutation, cross-over, or in the random initialization), it is also checked if the objectives of this point are dominated[1] by any element already stored in the archive. If not, the new point is added to the archive, and all points within the archive that are now dominated by the newly-added member are removed from the archive. Precaution has to be given in order to prevent an over-growing of the archive size (so-called "crowding"), but this is not of importance here, as we are focussing on running the algorithm in a smaller scale.

For learning more about EMO algorithms, the interested reader is referred to excellent text books [47][48], as well as the repository web site of Carlos Coello[2], and the survey article in a recent issue of this magazine [49].

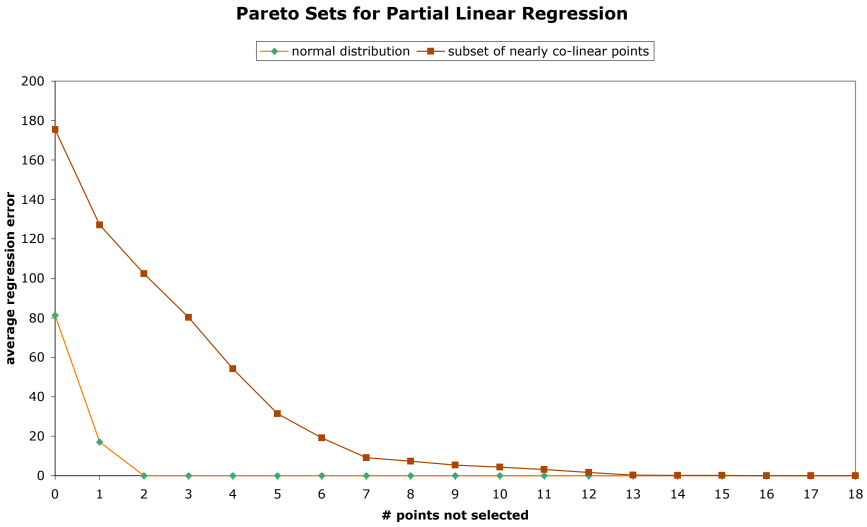


Figure 4: Complete PARETO fronts for the partial linear regression of 20 points. While in the first case all points are normally distributed around a line, in the second case, only a subset is nearly co-linear. It can be seen that both cases are very good distinguishable. As long as the selected subset in the second case does not contain any of the outliers, the average regression error will be very small. Remark: The PARETO sets here are discrete points. The points have been connected for illusration purposes only.

[1] Here, "dominates" stands for the relation between two points in objective space, where the dominating point is at least as good in all objectives as the dominated one, and better in at least one objective.

[2] http://www.lania.mx/ ccoello/EMOO/

After these few remarks on multi-objective optimization, and before continuing with the application of EMO for the partial linear regression problem, it is important to have a closer look onto two aspects of the problem domain: the above-stated multi-objective optimization problem has a well-defined PARETO front. If we represent an arbitrary *selection of a subset of points from a set of n points* by a *selection number* from the set $\{0, 1, 2, 3, \ldots, 2^n - 1\}$, the digits "1" in the representation of such a number as a dual number stand for the elements that are selected from the set of points. For example, if we have the set of three points $\{(1, 2), (3, 5), (6, 2)\}$, and the *selection number* $101_2 = 5_{10}$, the subset $\{(1, 2), (6, 2)\}$ is selected. For technical reasons, we have to exclude all selection numbers with only one digit being "1", as the linear regression needs at least two points. Also, selection number 0 is excluded. Among all possible selections of k elements from the given set of points, there will be one selection S_k with the smallest average regression error. The set $\{(k, S_k)|k = 2, \ldots, n\}$ then is the PARETO front of this multi-objective optimization problem.

Figure 4 shows a plot of the two completely sampled PARETO fronts for 20 points that were generated by two different randomized processes, with the second one including a subset of nearly co-linear points. It can be seen that in this case its very important to have *none* of the outliers in the selected subset, as otherwise the average regression error is rapidly increasing. If the selected subset only contains nearly co-linear points, the average regression error will have very small values, which is easily distinguishable from the reverse situation.

The second aspect of the problem domain that we have to take care for is the well-known curse-of-dimensionality. As it was already said in the introduction, in order to use a Tiny-GA, we should refrain from specifying the task in a "needle-in-a-haystack"-like manner. Probability issues of the problem domain can be easily seen. If we assume that our total set of n points is composed of m "good points" (the ones nearly located on the same line) and $n - m$ "bad points" (the outliers, the inclusion of which into the linear regression is disturbing the precision of the results), then we are looking for the probability of making a random selection that includes none of the "bad points." For all 2^n possible selections, there are exactly 2^{n-m} selections that will not include any of the m bad points, so the probability of a random guess is $1/2^m$. Its worth to note that this does not depend on n, so the probability of such a random selection does not depend on the relative amount of outliers, but on the absolute amount. So, we may expect that the EMO approach will only work up to a limited number of outliers.

Finally, we apply an EMO to the points given in fig. 2. The precise choice of the EMO algorithm itself does not matter, as we are not specifying a hard task (in this case, we have used the FPD-GA [50], with 10 individuals for 200 generations), and the evolving PARETO fronts of the archive can be seen in fig. 5. The algorithm has found a larger number of points in the feature space that show a rapid drop in the average regression error. Figure 6 gives a plot of the regression line represented by one of these individuals in the archive with a small average regression error.

3 Further Examples: Segmentation, Clustering and Classification

In this section, we will identify other domains, where the specification of a task for a Tiny-GAs is possible. We will see that all such tasks are subset-selection tasks, with a procedural criterion for the selection quality, and which can be easily encoded into a bitstring.

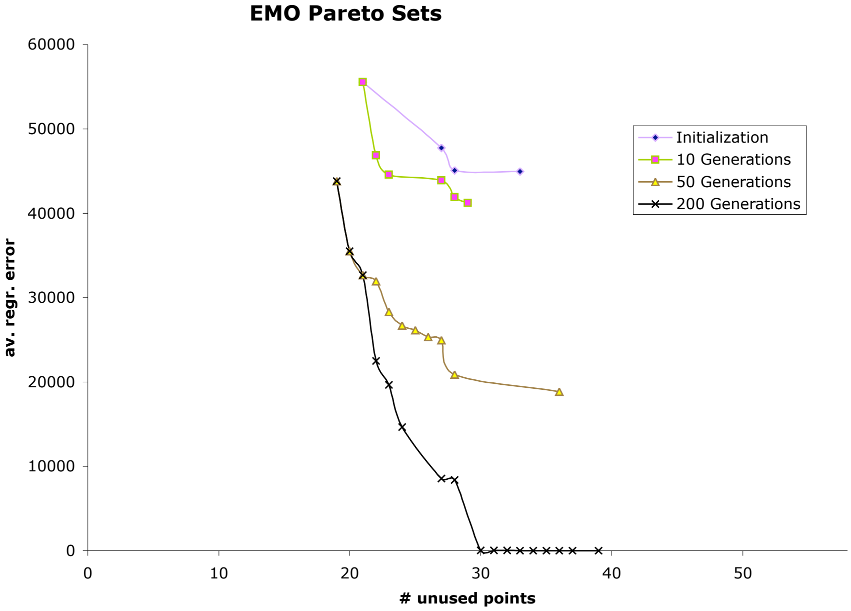


Figure 5: Objective points in the archive of the EMO algorithm after a number of generations. The points located very close to the bottom axis demonstrate the fact that in generation 200, the algorithm has already found a subset of points that does not contain any of the 21 outliers.

In the field of document layout recognition, such an evolutionary algorithm was already used to extract the invoice table part in the digitized image of an invoice sheet [51]. Several steps of preprocessing were needed to separate the various text lines that are present in the document image. A subset of these text lines is composing the invoice table itself. The approach was to select this subset by the internal similarity of the patterns of these extracted text stripes, and this was also achieved by a genetic algorithm running on a small scale.

The procedure was as follows: to each partial image of an extracted text row, a binary template was assigned. The image of a text row is divided into cells of equal width (the estimated width of a character in the image). Now, a bitstring is constructed from each such row, with the same size as the number of cells of a text row. The bit at position i in the bitstring is set to 1 if the corresponding text row cell i contains more black than white pixels (thus, there is a character of

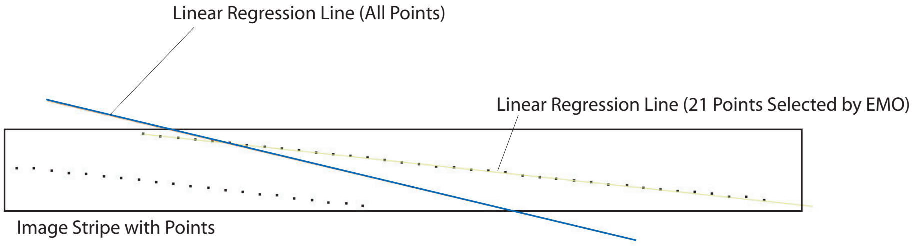

Linear Regression Line (All Points)
Linear Regression Line (21 Points Selected by EMO)
Image Stripe with Points

Figure 6: The regression lines for all 58 points, and for 21 points selected by one of the elements in the Pareto front.

text within), and 0 otherwise.

Doing this for each text row, the invoice sheet image becomes represented by a set of bitstrings. Among these bitstrings, we are interested now in the largest subset with highest similarity, and are focussing on using a genetic algorithm to help with the finding of such a subset.

In difference to the linear regression example that was studied in the foregoing section, here we do not consider the use of a second objective, but the use of *prototype* bitstrings. In the simplest case, this is just a bitstring of the same size as the set of bitstrings that represent the invoice sheet, and with having the smallest average Hamming distance to this set. It is not hard to find such a prototype bitstring.

The basic procedure to solve this problem is the so-called *Hamming fusion* of the set B of bitstrings derived from the text rows: with $B = \{b_i\}$ and $b_i[j]$ being the bit at position j in bitstring b_i, $H_j^0(B)$ annotates the number of occurrences of a 0 at position j:

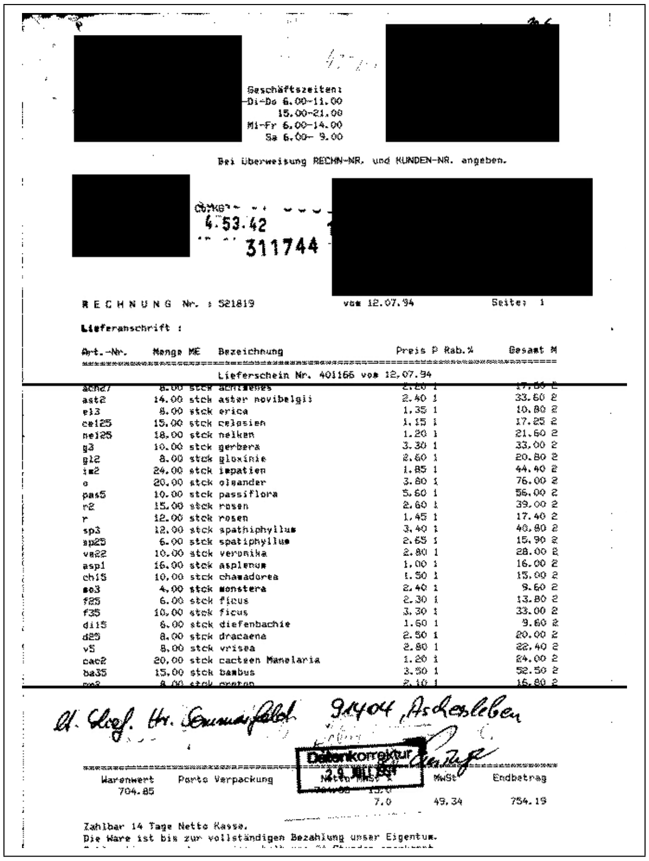
$$H_j^0(B) = |\{b_i \in B| \; b_i[j] = 0\}| \tag{2}$$

and $H_j^1(B) = |B| - H_j^0(B)$ gives the corresponding number of 1s at position j. Now, the Hamming fusion of B is given with the bitstring $t = HF(B)$ of length n and with $t[j] = 0$ if $H_j^0(B) > H_j^1(B)$, with $t[j] = 1$ if $H_j^0(B) < H_j^1(B)$ and either 0 or 1 if $H_j^0(B) = H_j^1(B)$. Obviously, the derivation of $HF(B)$ just counts for each position of the bitstrings whether there are more bits 0 or more bits 1, and takes the dominant bit at this position into the result. This is the bitstring with the smallest average Hamming distance to all bitstrings in B.

However, the study presented in [51] gave that just computing the Hamming fusion did not give sufficient results on a larger set of invoices: as can be seen in fig. 7, the field “Bezeichnung” contains item names that are differing in the number of characters. To account for such variations, the selection of *two or more* prototype bitstrings was considered, with each prototype only representing a fraction of the template bitstrings. One example is to use two prototypes, one for the left half of the template bitstrings, and one for the right half. Given two prototype bitstrings B_l and B_r of size $n/2$, their evaluation is as follows

1. For each template bitstring, compute the Hamming distance between B_l and the left half of bits of the template bitstring.
2. In case this Hamming distance exceeds a given threshold, the Hamming distance between B_r and the right half of bits of the template bitstring is computed.
3. Take the average of all computed Hamming distances, or a penalty term in case too few Hamming distances were computed.

A moment of reasoning gives that the Hamming fusion operation (e.g. of the left and right halfs of the template bitstrings) will not provide the optimal result. On the other hand, this procedure can be easily used for specifying a fitness function for a genetic algorithm - and that was actually



[redacted]

Geschäftszeiten:
Di-Do 6.00-11.00
15.00-21.00
Mi-Fr 6.00-14.00
Sa 6.00- 9.00

[redacted]

Bei Überweisung RECHN-NR. und KUNDEN-NR. angeben.

[redacted]

4.53.42
311744

[redacted]

RECHNUNG Nr. : 521819 vom 12.07.94 Seite: 1

Lieferanschrift :

Art.-Nr.	Menge	ME	Bezeichnung	Preis	P	Rab.%	Gesamt	M
			Lieferschein Nr. 401166 vom 12.07.94					
ach27	8.00	stck	achimenes	2.20	1		17.60	2
ast2	14.00	stck	aster novibelgii	2.40	1		33.60	2
e13	8.00	stck	erica	1.35	1		10.80	2
ce125	15.00	stck	celosien	1.15	1		17.25	2
ne125	18.00	stck	nelken	1.20	1		21.60	2
g3	10.00	stck	gerbera	3.30	1		33.00	2
gl2	8.00	stck	gloxinie	2.60	1		20.80	2
im2	24.00	stck	impatien	1.85	1		44.40	2
o	20.00	stck	oleander	3.80	1		76.00	2
pas5	10.00	stck	passiflora	5.60	1		56.00	2
r2	15.00	stck	rosen	2.60	1		39.00	2
r	12.00	stck	rosen	1.45	1		17.40	2
sp3	12.00	stck	spathiphyllus	3.40	1		40.80	2
sp25	6.00	stck	spatiphyllus	2.65	1		15.90	2
ve22	10.00	stck	veronika	2.80	1		28.00	2
asp1	16.00	stck	asplenum	1.00	1		16.00	2
ch15	10.00	stck	chamadorea	1.50	1		15.00	2
mo3	4.00	stck	monstera	2.40	1		9.60	2
f25	6.00	stck	ficus	2.30	1		13.80	2
f35	10.00	stck	ficus	3.30	1		33.00	2
di15	6.00	stck	diefenbachie	1.60	1		9.60	2
d25	8.00	stck	dracaena	2.50	1		20.00	2
v5	8.00	stck	vrisea	2.80	1		22.40	2
cac2	20.00	stck	cacteen Manelaria	1.20	1		24.00	2
ba35	15.00	stck	bambus	3.50	1		52.50	2
cro2	8.00	stck	croton	2.10	1		16.80	2

d. Lief. Hr. Sommerfeld 91404 Aschersleben

Datenkorrektur
29. JULI 1994

Warenwert	Porto	Verpackung	Netto	MwSt %	MwSt	Endbetrag
704.85			704.85	15.0		
				7.0	49.34	754.19

Zahlbar 14 Tage Netto Kasse.
Die Ware ist bis zur vollständigen Bezahlung unser Eigentum.

Figure 7: Example for a scanned image of an invoice sheet. The two thick vertical lines represent the invoice table part that has been extracted from the document by using a Tiny-GA.

done in the invoice sheet processing. After some test runs, a standard genetic algorithm with 10 individuals run for 50 generations. If the best fitness value in the population was above 0.7, the result was accepted, and all text rows similar to this prototype (also a thresholded Hamming distance) were selected. In [51], it was reported that such a procedure could succesfully select close to 90 % of the text stripes. Due to the low computational complexity of the fitness function, and the small population and generation count needed, this application of a "Tiny-GA" could also be used online.

Going further, we may find a similar approach to a much more general problem: the *robust* clustering of data. With the attribute "robustness" we want to refer to the stability of the clustering result, once the data values undergo smaller changes, or new data values are added. Among the huge selection of clustering methods, and for a given, yet not explored set of data, we may consider some of the clustering methods as *appropriate* and some of them as *inappropriate*. The meaning here, in other words, is as: when a clustering method is inappropriate for the given data set, it will produce stronger differing clustering results when the initial conditions or the given data values are randomly modified, than for an appropriate clustering method (see fig. 8).

Assume a (numbered) data set $d = (d_1, d_2, \ldots, d_n)$, which has to be separated into two clusters C_1 and C_2. Further assume a set of $M_1, M_2, \ldots, M_k$ clustering algorithms (e.g. k-means, neural gas,...), the $M_1(c_i), M_2(c_i), \ldots, M_k(c_i), 1 \le i \le 2$, results of which, when applied to d, will depend

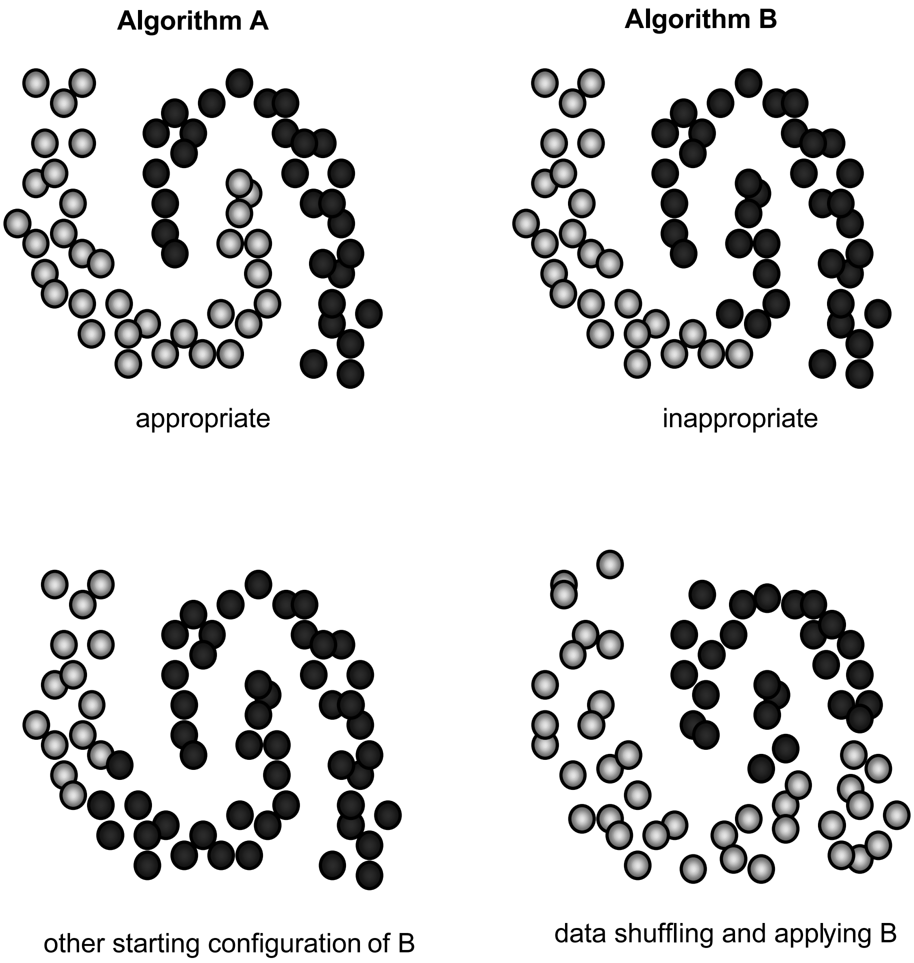

Algorithm A
Algorithm B
appropriate
inappropriate
other starting configuration of B
data shuffling and applying B

Figure 8: Scheme for applying *appropriate* or *inappropriate* clustering methods to unknown data.

on an initial configuration of the c_i centers, and a set V of l random modifications of d, e.g. moving the data point by a small amount in a random direction. Then, when a clustering algorithm $M_p, 1 \leq p \leq k$, is applied to the modified data set $V_j(d)$, it will assign each data point in d to either C_1 or C_2. This can be represented by a bitstring. For example, if there are four data points, the bitstring 0110 describes that data point d_1 and d_4 have been assigned to cluster C_1 and data points d_2 and d_3 have been assigned to cluster C_2.

Applying all k clustering algorithms to all l modifications of the data set d will give $k \cdot l$ bitstrings. Under the assumption given above, for appropriate clustering methods, there should be more similar bitstrings than for inappropriate ones. In other words: the most common scheme among all $k \cdot l$ bitstrings hints on an appropriate clustering method.

Again, it seems that we have the Hamming fusion as an appropriate procedure to yield the prototype of the most typical clustering result for the given data. Again, there is a problem: for the clustering algorithm, it does not make a difference if it assigns the cluster number 1 or 2 to an element of the data set, as long as it assigns different cluster numbers to elements from different clusters. For the bitstrings, it makes a difference, as the “interpretation” of all bitstrings representing cluster results has to equalize a bitstring and its reverse.

Since it is simple to account for this in the fitness specification of a genetic algorithm (find the prototype, to which the average Hamming distance of each cluster result bitstring, or to its inverse, is minimal), usage of a Tiny-GA in this context seems straightforward.

Finally, we also want to consider the fitness values itself obtained after a few generations of a small evolutionary algorithm for the purpose of classification. Object classes can be specified according to intervals, into which these fitness value fall.

An example for such a "fitness calibration" was given in [52]. There, the issue of Kirlian images was considered, and the task was to study whether Kirlian images reflect some kind of intra-person specificity or not. Kirlian images from all fingers were taken from a number of subjects at different times, and subjects were selected with perceptually similar Kirlian patterns. Now, a Tiny-GA was used to *measure* that perceptual similarity.

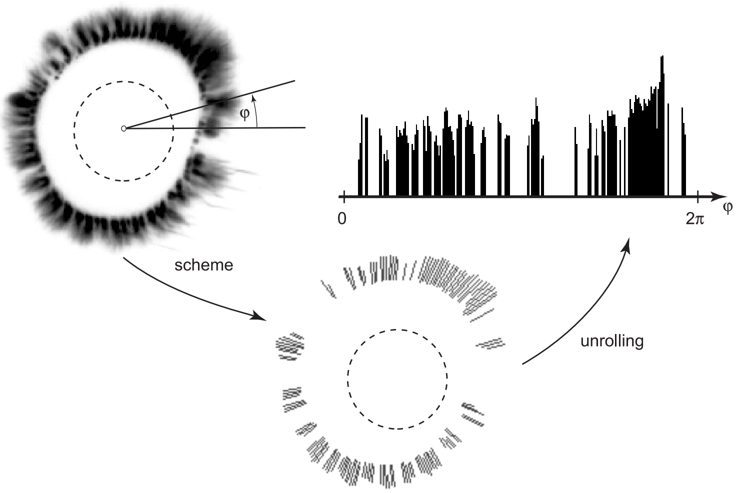
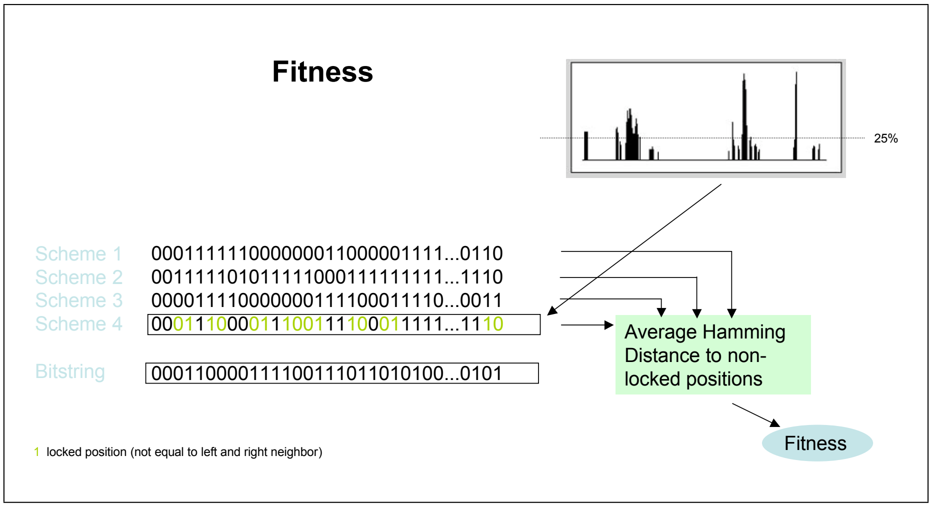


Figure 9: Unrolling of a Kirlian image.

First, each finger image was unrolled (see fig. 9). The unrollings were used to define the data bitstrings: if at angle ϕ the fraction of the beam in the direction ϕ and starting from the (interactively set) center C of the Kirlian image, is larger than 25% of the maximum value in any direction, the bit was set to 1, 0 otherwise. By this procedure, each group of perceptual similar Kirlian images gave a set of four bitstrings. As in the cases before, a prototype bitstring was searched with the smallest average Hamming distance to all four bitstrings, where the Hamming distance is only taken among bit positions, which are equal to their neighbors. But instead of using Hamming fusion (that does not solve the problem anyway, as some bit positions are excluded from the distance measurement), this restricted average Hamming distance was used as fitness function, and a standard genetic algorithm run for 200 generations, to yield the fitness value of the best individual obtained after these generations (and before the algorithm reaches the optimum).

From these fitness values, three classes could be separated: four random patterns gained fitness values between 0.25 and 0.30; randomly selected sets of Kirlian images gained fitness values between 0.28 and 0.32; and all perceptual groups of Kirlian images gained fitness values between 0.34 and 0.53. Thus, the class of perceptual similar Kirlian images of the same subject is clearly



Fitness

Scheme 1 0001111111000000001100000111 1...0110
Scheme 2 0011111010111111000111111111...1110
Scheme 3 0000111110000000011110001111 0...0011
Scheme 4 0001110000111001111100011111...1110

Bitstring 0001100001111001110110101 00...0101

Average Hamming Distance to non-locked positions

Fitness

1 locked position (not equal to left and right neighbor)

Figure 10: Processing of a prototype bitstring: the four schemes of four KIRLIAN images are extracted from the unrolled KIRLIAN patterns. Only positions that are equal to their neighbors in the bitstrings are considered for computing the average Hamming distance of a given bitstring to these four schemes. This value then is used as a fitness of the bitstring.

distinguished from the class of random KIRLIAN images and random patterns.

The former examples have shown that the use of Tiny-GAs in applications is related to some subset-selection theme, be it the selection of a larger subset of points establishing a high value of a linear regression, be it the extraction of a larger subset of template bitstrings of highest similarity, or the same selection for various two-cluster results accounting for the possible inversion of the bitstrings that represent the results. The underlying mathematical problem, referred to as *Generalized Bitstring Prototype Problem* (GBPP), was studied in [53], showing that in only a small number of cases there is an exact solution that can be explicitely computed from the bitstrings (as it can be done for the Hamming fusion). However, the study also left many questions open that might be worth a further exploration.

Given the straightforward encoding of the GBPP into bitstrings, its hardness, and the simple fitness computation, an identification of such a problem in the domain of an image processing framework favours the use of a Tiny-GA.

4 Conclusions

Image processing, despite of comprising a challenging domain of computer science, which includes mass-data processing and also demands low computational effort, can nevertheless benefit from the usage of simple evolutionary algorithms. One pre-requisite for this is a proper structuring of the image processing framework, as well as the identification of subset-selection tasks within the framework modules. Examples for such subset-selection tasks were given in this paper. These tasks, then, can be handled by most evolutionary algorithms with small populations and within a few generations. This does not contradict with the more common use of evolutionary computation, which deals with problems of high complexity. But sometimes, a side-glance to the runtime characteristics of evolutionary algorithms in domains of much smaller complexity may be worth the extra conceptual effort.

References

[1] D.E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, Reading MA, 1989.

[2] J.H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

[3] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4:65–85, 1994.

[4] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[5] J.R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press, Cambridge, London, 1994.

[6] I. Rechenberg. *Evolutionsstrategie.* Frommann-Holzboog, Stuttgart, 1994. in German.

[7] J.T. Alander. An indexed bibliography of genetic algorithms in signal and image processing. Technical Report 94-1-SIGNAL DRAFT May 18, 1999, University of Vaasa, 1999.

[8] D. Van de Ville, M. Nachtegael, D. Van der Weken, E. E. Kerre, W. Philips, and I. Lemahieu. Noise reduction by fuzzy image filtering. *IEEE Transactions on Fuzzy Systems*, 11(4):429 – 436, 2003.

[9] E. Jones, P. Runkle, N. Dasgupta, L. Couchman, and L. Carin. Genetic algorithm wavelet design for signal classification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(8):890–895, August 2001.

[10] P. Kraft, N.R. Harvey, and S. Marshall. Parallel genetic algorithms in the optimization of morphological filters: A general design tool. *Journal of Electronic Imaging*, 6(4):504–516, October 1997.

[11] I. Yoda, K. Yamamoto, and H. Yamada. Automatic acquisition of hierarchical mathematical morphology procedures by genetic algorithms. *Image and Vision Computing*, 17(10):749–760, August 1999.

[12] T. Nagao and S. Masunaga. Automatic construction of image transformation processes using genetic algorithm. In *Proc. ICIP96*, pages 3:731–734, 1996.

[13] S. Aoki and T. Nagao. Automatic construction of tree-structural image transformation using genetic programming. In *Proc. ICIP99*, pages I:529–533, 1999.

[14] D. Saupe and M. Ruhl. Evolutionary fractal image compression. In *Proceedings IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 129 – 132, 1996.

[15] S.K. Mitra, C.A. Murthy, and M.K. Kundu. Technique for fractal image compression using genetic algorithm. *IEEE Trans. Image Processing*, 7(4):586–593, April 1998.

[16] M. Takezawa, H. Honda, M. Hasegawa, and H. Kitajima. A genetic-algorithm based quantization method for fractal image coding. In *Proc. ICIP99*, pages I:458–461, 1999.

[17] A.S. Abutaleb and M. Kamel. A genetic algorithm for the estimation of ridges in fingerprints. *IEEE Trans. Image Processing*, 8(8):1134–1139, August 1999.

[18] K. Franke. *The Influence of Physical and Biomechanical Processes on the Ink Trace - Methodological foundations for the forensic analysis of signatures.* PhD thesis, Artifical Intelligence Institute, University of Groningen, The Netherlands, 2005.

[19] K. Franke, J. Ruiz-del-Solar, and M. Köppen. Soft biometrics: Soft computing for biometric applications. *International Journal of Fuzzy Systems*, 4(2):665 – 672, 2002.

[20] B. Bhanu and S. Lee. *Genetic Learning for Adaptive Image Segmentation.* Kluwer, 1994.

[21] P. Andrey and P. Tarroux. Unsupervised image segmentation using a distributed genetic algorithm. *Pattern Recognition*, 27(5):659–673, May 1994.

[22] C.T. Li and R. Chiao. Multiresolution genetic clustering algorithm for texture segmentation. *Image and Vision Computing*, 21(11):955–966, October 2003.

[23] D. Maio, D. Maltoni, and S. Razzi. Topological clustering of maps using a genetic algorithm. *Pattern Recognition Letters*, 16:89–96, 1995.

[24] B.K. Jeon, J.H. Jang, and K.S. Hong. Road detection in spaceborne SAR images using a genetic algorithm. *IEEE Trans. Geoscience and Remote Sensing*, 40(1):22–29, January 2002.

[25] C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, November 2004.

[26] L.S. Oliveira, R. Sabourin, F. Bortolozzi, and C.Y. Suen. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In *Proc. ICPR02*, pages I: 568–571, 2002.

[27] I.S. Oh, J.S. Lee, and B.R. Moon. Hybrid genetic algorithms for feature selection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(11):1424–1437, November 2004.

[28] M.A. Kupinski and M.A. Anastasio. Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curves. *IEEE Trans. Medical Imaging*, 18(8):675–685, August 1999.

[29] C. de Stefano, A. della Cioppa, and A. Marcelli. Exploiting reliability for dynamic selection of classifiers by means of genetic algorithms. In *Proc. ICDAR03*, pages 671–675, 2003.

[30] K. Sirlantzis and M.C. Fairhurst. Optimisation of multiple classifier systems using genetic algorithms. In *Proc. ICIP01*, pages I: 1094–1097, 2001.

[31] J.J. Jacq and C. Roux. Registration of 3-d images by genetic optimization. *Pattern Recognition Letters*, 16:823–841, 1995.

[32] P.W.M. Tsang and Z. Yu. Genetic algorithm for model-based matching of projected images of three-dimensional objects. *IEE Proceedings-Vision Image and Signal Processing*, 150(6):351–359, December 2003.

[33] F. Samadzadegan, A. Azizi, M. Hahn, and C. Lucas. Automatic 3d object recognition and reconstruction based on neuro-fuzzy modelling. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(5):255–277, August 2005.

[34] L. Luo, D. Clewer, N. Canagarajah, and D.R. Bull. Genetic stereo matching using complex conjugate wavelet pyramids. In *Proc. ICIP01*, pages II: 153–156, 2001.

[35] M. Gong and Y.H. Yang. Genetic-based stereo algorithm and disparity map evaluation. *International Journal of Computer Vision*, 47(1-3):63–77, April 2002.

[36] S.M. Bhandarkar, Y.Q. Zhang, and W.D. Potter. An edge-detection technique using genetic algorithm-based optimization. *Pattern Recognition*, 27(9):1159–1180, September 1994.

[37] M. Gudmundsson, E.A. Elkwae, and M.R. Kabuka. Edge-detection in medical images using a genetic algorithm. *IEEE Trans. Medical Imaging*, 17(3):469–474, June 1998.

[38] Q. Ji and Y. Zhang. Camera calibration with genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics A*, 31(2):120–130, March 2001.

[39] S. Hati and S. Sengupta. Robust camera parameter estimation using genetic algorithm. *Pattern Recognition Letters*, 22(3-4):289–298, March 2001.

[40] A.D.J. Cross and E.R. Hancock. Recognizing building patterns using matched-filters and genetic search. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53(2):95–107, April 1998.

[41] J. Bala, K. DeJong, J. Huang, H. Vafaie, and H. Wechsler. Visual routine for eye detection using hybrid genetic architectures. In *Proc. ICPR96*, volume C, pages 606–610, 1996.

[42] C.S. Shieh, H.C. Huang, F.H. Wang, and J.S. Pan. Genetic watermarking based on transform-domain techniques. *Pattern Recognition*, 37(3):555–565, March 2004.

[43] R.Z. Wang, C.F. Lin, and J.C. Lin. Image hiding by optimal LSB substitution and genetic algorithm. *Pattern Recognition*, 34(3):671–683, March 2001.

[44] Y. Wang and N. Funakubo. Detection of geometric shapes by the combination of genetic algorithm and subpixel accuracy. In *Proc. ICPR96*, volume D, pages 535–539, 1996.

[45] H.C. Shyu and J.J. Leou. Detection and concealment of transmission errors in mpeg-2 images–a genetic algorithm approach. *IEEE Trans. Circuits and Systems for Video Technology*, 9(6):937, September 1999.

[46] J.D. Knowles, D.W. Corne, and M. Fleischer. Bounded archiving using the Lebesgue measure. In *Evolutionary Computation 2003. CEC03. The 2003 Congress on Evolutionary Computation*, pages 2490–2497, 2003.

[47] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, 2001.

[48] C.A. Coello Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2002.

[49] C.A. Coello Coello. Evolutionary multi-objective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1), 2006.

[50] M. Köppen, K. Franke, and B. Nickolay. Fuzzy-pareto-dominance driven multiobjective genetic algorithm. In *Proceedings of the 10th IFSA World Congress. Istanbul, Turkey, 2003*, pages 450–453, 2003.

[51] M. Köppen, D. Waldöstl, and B. Nickolay. A system for the automated evaluation of invoices. In Jonathan H. Hull and Suzanne L. Taylor, editors, *Document Analysis Systems II*, pages 223–241. World Scientific, Singapore a.o., 1997.

[52] H. Treugut M. Köppen, B. Nickolay. Genetic algorithm based heuristic measure for pattern similarity in kirlian photographs. In E.J. Boers, editor, *Applications of Evolutionary Computing*, Springer Lecture Notes on Computer Science 2037, pages 317–324, 2001.

[53] M. Köppen and E. Dimitriadou. Concurrent application of genetic algorithm in pattern recognition. In A. Abraham, M. Köppen, and K. Franke, editors, *Design and Application of Hybrid Intelligent Systems*, Frontiers in Artificial Intelligence and Applications Vol. 104, pages 868–877. IOS Press Amsterdam, Berlin, Oxford, Tokyo, Washington D.C., 2003.