International Journal of Document Analysis and Recognition (IJDAR) manuscript No.
(will be inserted by the editor)

Christian Gagné · Marc Parizeau

Genetic Engineering of Hierarchical Fuzzy Regional Representations for Handwritten Character Recognition

Received: February 15th, 2005 / Revised version: July 25th, 2005

Abstract This paper presents a genetic programming based approach for optimizing the feature extraction step of a handwritten character recognizer. This recognizer uses a simple multilayer perceptron as a classifier and operates on a hierarchical feature space of orientation, curvature, and center of mass primitives. The nodes of the hierarchy represent rectangular sub-regions of their parent node, the tree root corresponding to the character's bounding box. Within each sub-region, a variable number of fuzzy features are extracted. Genetic programming is used to simultaneously learn the best hierarchy and the best combination of fuzzy features. Moreover, the fuzzy features are not predetermined, they are inferred from the evolution process which runs a two-objective selection operator. The first objective maximizes the recognition rate, and the second minimizes the feature space size. Results on Unipen data show that, using this approach, robust representations could be obtained that out-performed comparable human designed hierarchical fuzzy regional representations.

Keywords On-line character recognition – Handwriting – Evolutionary computations – Fuzzy logic – Unipen dataset.

1 Introduction

Pattern recognition systems are classically modeled as a processing pipeline made up of raw input sensing, segmentation, feature extraction, classification, and post-processing [7]. Except for classification where many well-known generic methods exist, all of these steps are mostly problem specific. The design of a recognition system thus requires a thorough understanding of the recognition task and, most often, involves some form of a trial and error process before adequate system performance can be reached. In particular, the segmentation and feature extraction steps are very critical because no matter how powerful a classifier may be, it can never fully compensate for noisy or non-discriminating features.

This paper is about the segmentation and feature extraction components of an on-line handwritten character recognition system [20], and on how the development of these two steps can be partly automated using Genetic Programming (GP) [3,14]. GP is a machine intelligence technique involving the simulation of natural evolution for the automatic programming of computers. It is a generic problem solving method applicable whenever solutions can be represented by a computer program and evaluated by an objective function; the so-called "fitness" function. Populations of programs – initially random programs – evolve over time through a sequence of processes that include (natural) selection, crossover operations to exchange genetic material between two programs, and mutation operations to randomly modify parts of the evolved programs. In the end, the fittest individual (program) is chosen as "the" solution to the problem and, although GP systems do not guarantee convergence to an optimal solution, they have been shown in practice to outperform other techniques as well as human experts for several hard problems [15].

In the context of character recognition, the output of the feature extraction module, i.e. the character representation, is possibly the most important key for the development of higher performance systems [29]. In this paper, we focus on a specific on-line representation and, using GP, we try to improve upon it with techniques that involve minimal human intervention. The organization of the paper is as follows. In Section 2, the original "fuzzy-regional" handwriting representation [13] is first summarized and previously obtained results are recalled for baseline comparison purposes. Results are also given in this section for a new hierarchical variation of the original uniform grid configuration. Then, Section 3 introduces a new data-driven approach where the hier-

This work was supported by NSERC-Canada.

The authors are with the Laboratoire de Vision et Systèmes Numériques (LVSN), Département de Génie Électrique et de Génie Informatique, Université Laval, Québec (Québec), Canada, G1K 7P4. Contact author: parizeau@gel.ulaval.ca

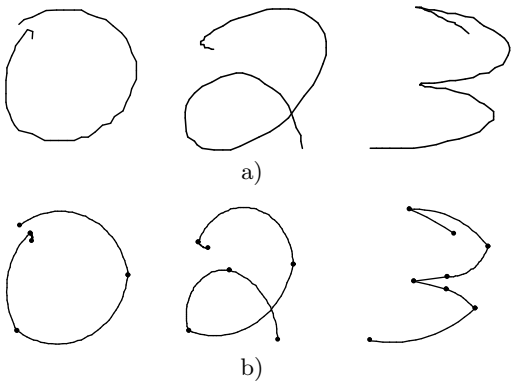

Fig. 1 a) Three digits; b) their circular arc stroke reconstruction.

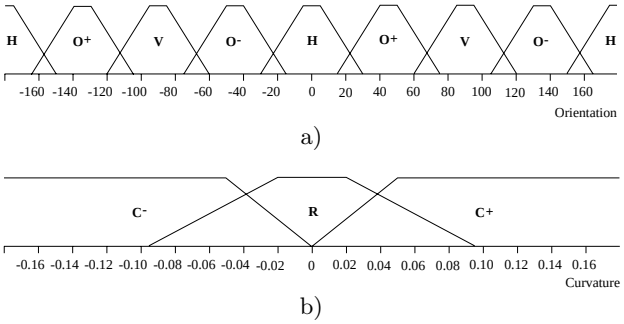

Fig. 2 a) Sets **H**, **V**, **O**+, and **O**− for parameter θ; b) sets **R**, **C**+, and **C**− for parameter c.

archical grid segmentation of the character is no longer static, but now depends on the spatial distribution of handwriting strokes. Next, in Section 4, the method for finding the optimal genetically engineered hierarchical representation is presented, together with experimental results. Finally, Section 5 concludes the paper with a discussion on how its actual contributions can be generalized for the design of feature extraction components for other kinds of pattern recognition systems.

2 Fuzzy-Regional Representations

The on-line fuzzy-regional representation [13,18] starts with a stroke decomposition approximated by a sequence of circular arcs, as described in [17]. A character is thus represented by a sequence of circular arc strokes

$$s_1, s_2, \ldots, s_i, \ldots, s_q,$$

where each arc $s_i = (\mathbf{p}_0, \mathbf{p}_1, l, c)$ is described by four parameters: $\mathbf{p}_0$ and $\mathbf{p}_1$ are respectively the starting and ending points of the arc, l is its curvilinear length, and c its curvature. A stroke orientation θ is also determined from the angle of vector $\overrightarrow{\mathbf{p}_0\mathbf{p}_1}$. Figure 1 shows some examples of isolated digits and their circular arc stroke reconstruction. The circular arc segmentation algorithm is based on the local extrema and inflection points of the on-line curvature signal.

2.1 Feature Extraction

The feature extraction module decomposes the character into a fixed number of rectangular regions in the character's bounding box. Within each of these regions, a fuzzy vector is extracted from the orientation and curvature of the strokes. Figure 2 shows the fuzzy sets that are used to quantify the segmented strokes. The four characteristic functions of Figure 2a enable the fuzzyfication of the stroke angle $\theta \in [-180, 180]$, and the three functions of Figure 2b are for the stroke curvature $c \in [-\infty, \infty]$. A fuzzyfied stroke can thus be considered a more or less horizontal (**H**), vertical (**V**), oblique with positive slope (**O**+), or oblique with negative slope (**O**−), depending on the stroke angle θ. Similarly, a stroke can be more or less rectilinear (**R**), have a positive curvature (**C**+) or a negative curvature (**C**−). As only one fuzzy vector is associated to each region of a character, the maximum fuzzy membership value in each fuzzy set is considered when several strokes occur in the same region. For example, if one region contains two strokes, one very **C**+ (thus not very **C**−) and the other very **C**− (thus not very **C**+), then the resulting fuzzy vector will be both very C+ and very C-. The complete feature vector space is constructed by simple concatenation of the different regional fuzzy vectors. This fuzzyfication process allows the mapping of different regional characteristics extracted from a variable number of strokes into a fixed-length vector with values in the $[0, 1] \in I\!R$ interval.

A problem with the previous representation is that stroke orientation is global and does not give precise information about the local orientation of the strokes within each region. A refinement proposed originally in [13] is to crop the strokes at the region boundaries before computing the θ angles. Thus, the end points used for computing this orientation angle are never outside the region boundaries. Another refinement was to normalize each fuzzy membership by multiplying it with the ratio of the length of the associated stroke over the diagonal length of the region. These two refinements are used throughout the paper for every experiment.

In addition to the previous seven fuzzy variables, two new features are also extracted from each region: the horizontal (**MX**) and vertical (**MY**) centers of mass of the strokes after cropping. These new features are also normalized within the $[0, 1]$ interval, and represent the coordinates of a relative position in the region extent; $(0, 0)$ is the lower left corner of the region, and $(1, 1)$ its upper right corner.

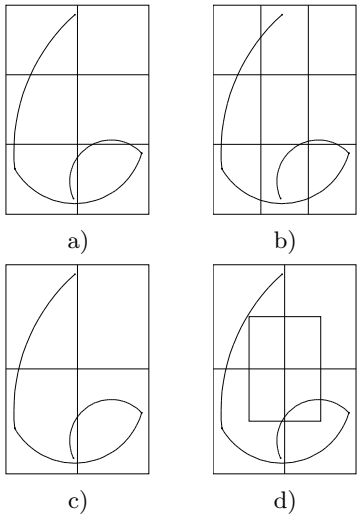

Fig. 3 Different grid topologies: a) regular 3×2 (RFR-3x2); b) regular 3×3 (RFR-3x3); c) 2-level quad tree (RFR-quad2); d) 2-level quin tree (RFR-quin2).

2.2 Grid Topologies

In previous papers [13,18], it was proposed to divide the character bounding box into a uniform grid with, for example, 3×2 or 3×3 regions, as shown respectively in Figure 3a and 3b. In this paper, it is also proposed to use a hierarchical grid topology, starting from the character bounding box, subdividing the region into four regions (quad tree) or five regions (quin tree), recursively. Figure 3c shows the five region decomposition for a 2-level quad tree, while Figure 3d shows the six region decomposition for a 2-level quin tree. 3-level quad tree (RFR-quad3) and 3-level quin tree (RFR-quin3) representations are also tested as standard grid topologies. These hierarchical quad and quin tree representations are inspired from the work of Park et al. [19].

2.3 Experimental Results

All classification results presented in this paper have been obtained using a MultiLayer Perceptron (MLP) classifier trained with standard on-line back-propagation and momentum [12], with a single hidden layer of 50 neurons. The learning rate and momentum are fixed respectively at 0.1 and 0.25. Training was controlled using a cross-validation procedure where 67% of the learning set is used for training and 33% for validation. The MLP was trained for a minimum of 35 epochs and the total number of training epochs varied from 65 to 125. The decision strategy (the post-processing module) is simply to classify data according to the maximum output of the MLP (no rejection).

Experiments were conducted on Section 1a (isolated digits) of the Unipen data set [11]. Training data are from Unipen Train-R01/V07, consisting in 15 953 characters, while testing data are from DevTest-R02/V02, consisting of 8 598 characters. Experiments are conducted using the complete data sets, except for those character samples that have zero width and zero height, as reported in [18] (4 in the training set and 34 in testing set).

The recognition rates obtained for different human-engineered topologies are reported in Table 1. Experiments are conducted on the six standard grid topologies presented in Section 2.2, extracting for each region the nine features described in Section 2.1. Thus, the feature set size for each topology is the number of regions multiplied by nine. The "Mean Rec. Rate" column is the average recognition rate on the test set for ten different training runs of MLPs, the "Max. Rec. Rate" column is the best recognition rate achieved over these ten runs, and the "Stdev. Rec. Rate" is the standard deviation around the average. The "Mean Shift RFR-3x2" and "Max. Shift RFR-3x2" columns are respectively the average and maximum recognition rate improvements over the mean rate of RFR-3x2 which serves as a baseline for this study.

Results show an average recognition rate of 95.60% for the baseline RFR-3x2. From this baseline representation, improvements are achieved mostly by representations with many more features (up to five times): +0.25% for RFR-3x3, +0.73% for RFR-quad3 and +0.77% for RFR-quin3. On the other hand, the 2-level quad and quin tree representations show a −0.24% decrease in performance for RFR-quin2 and −0.87% for RFR-quad2. These results demonstrate that the grid topology may have an important effect on performance using such regional representations. The RFR-quin3 topology performs the best with an improvement of +0.77%, on average, and with a best case scenario of +1.06%. The RFR-quad3 performs a little worse than RFR-quin3 on average (+0.73% vs +0.77%), but this may not be statistically significant. However, it uses 90 less features than its RFR-quin3 counterpart, which is a significant advantage.

A direct comparison of the previous results with those found in the literature is not easy. In a survey compiled by Ratzlaff [23], it seems that most recognition rates reported for Section 1a of Unipen were obtained using subsets of Train-R01/V07, not with DevTest-R02/V02 which appears to be significantly harder. In reference [18], we have reported an average recognition rate of up to 96.9% (compared with 96.37% for the best result of Table 1) using similar regional representations, but with additional global features. To the best of our knowledge, the only other report for the complete DevTest-R02/V02 was made by Ratzlaff who obtained 98.1% using a different representation and a different classifier [22]. But he also states in his paper that he was forced to manually label the data as he did not have access to the official

Grid Topology	Feature Set Size	Mean Rec. Rate (%)	Max. Rec. Rate (%)	Stdev. Rec. Rate (%)	Mean Shift RFR-3x2	Max. Shift RFR-3x2
RFR-3x2	54	95.60	95.77	0.18	–	+0.18
RFR-3x3	81	95.84	96.09	0.21	+0.25	+0.49
RFR-quad2	45	94.72	95.07	0.25	−0.87	−0.52
RFR-quad3	189	96.33	96.66	0.17	+0.73	+1.06
RFR-quin2	54	95.35	95.65	0.20	−0.24	+0.05
RFR-quin3	279	96.37	96.66	0.21	+0.77	+1.06

Table 1 Recognition rates for different grid topologies of fuzzy-regional representations.

class labels[1]. Thus, it is probable that this manual relabeling of DevTest-R02/V02 has cleaned up the data to some extent, possibly correcting any mislabeling. Moreover, through a recognition experiment conducted with 10 human readers, it has been shown previously [13] that about 1% of the digits found in section 1a of DevTest-R02/V02 are totally unrecognizable. This suggests that this 1% of characters may be mislabeled.

In any case, the aim of this paper is not to develop an optimal recognition system per se, as this would probably imply the use of multiple classifiers and representations, but rather to explore different topologies in the context of region based fuzzy representations, and to experiment with GP for genetically engineering such topologies.

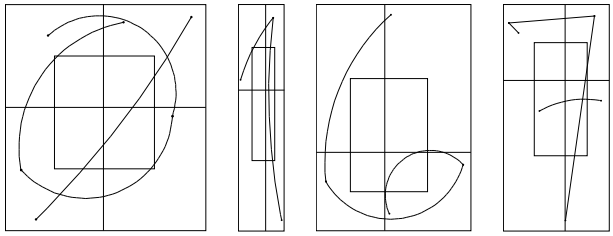

Fig. 4 2-level quin tree data-driven grid topology (DDR-quin2) for different digit characters: a) a zero; b) a one; c) a six; d) a seven.

3 Data-driven Representations

The fuzzy-regional representation has several interesting aspects, such as the mapping of an arbitrary online script into a fixed-length normalized feature vector. On the other hand, the use of a static grid topology may be sub-optimal, especially for deformed or slanted characters. Inspired by the work of Park et al. [19], we now study the use of data-driven hierarchical topologies where regions are recursively defined around the center of mass of strokes, instead of the absolute center of the parent region. For example, in the case of the quin tree topology, the four corners of the central region correspond to the centers of the four other regions, which are themselves defined by the center of mass of the whole character, as illustrated by Figure 4. For the 3-level quad and quin tree representations, the position of each region at the third level is computed using the center of mass of its parent region at the second level. For the data-driven 3x2 and 3x3 grid topologies, the width and height of each region are adjusted according to the center of mass of the character relative to its geometrical center.

For all data-driven topologies, because centers of mass are already used to determine the region boundaries, the corresponding features (**MX** and **MY**) are removed from the fuzzy representation, and replaced by a region height/width ratio (**HW**). The final representation length is thus 8 times the total number of regions.

3.1 Experimental Results

Table 2 shows the recognition results for the data-driven representations, with experiments conducted in the same conditions as those exposed in Section 2.3. Results show that, contrary to expectations, data-driven representations mostly tend to decrease performance, especially for the quad and quin tree topologies. This illustrates the fact that feature extraction can be counterintuitive. Only the DDR-3x3 seems to benefit a little from the data-driven approach, up to +0.51% better than the baseline, and +0.26% better than its RFR equivalent. Figure 5 gives a bar graph comparing the mean recognition rate shift over the baseline for the different RFR and DDR representations.

4 Genetical Engineering of Representations

Up until now, a fixed set of predetermined hand-crafted static and data-driven topologies were tested, and we have shown experimentally that some are better than others. Also, a fixed set of features was systematically extracted from every region, sometimes generating large representations that may suffer from the well-known curse of dimensionality. Finally, the features themselves have been predefined by a fixed set of hard-coded characteristic functions. The objective of this section is to establish an automatic procedure based on genetic programming

[1] DevTest-R02/V02 was originally released to Unipen contributors without labels in preparation for a competition which was finally never organized. Later on, the labels were released in a separate file hierarchy which needed to be merged with the original unlabeled data.

Grid Topology	Feature Set Size	Mean Rec. Rate (%)	Max. Rec. Rate (%)	Stdev. Rec Rate (%)	Mean Shift RFR-3x2	Max. Shift RFR-3x2
DDR-3x2	48	95.43	95.62	0.12	−0.17	+0.03
DDR-3x3	72	96.10	96.29	0.12	+0.51	+0.69
DDR-quad2	40	93.98	94.21	0.21	−1.62	−1.39
DDR-quad3	168	94.97	95.31	0.23	−0.63	−0.29
DDR-quin2	48	95.02	95.26	0.18	−0.58	−0.34
DDR-quin3	248	94.69	95.09	0.21	−0.91	−0.51

Table 2 Recognition rates for different grid topologies of data-driven representations.

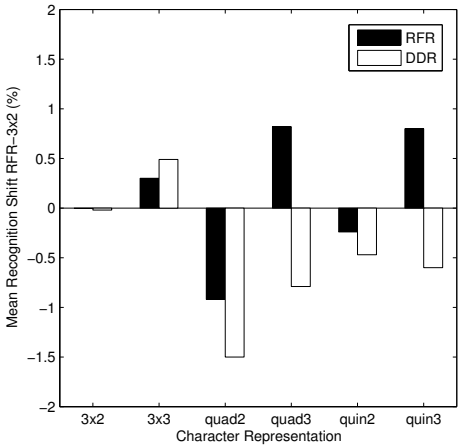

Fig. 5 Mean recognition rate shift over the basic RFR-3x2 for tested RFR and DDR.

to explore more diverse topologies with per region specialized features.

In previous work, several papers have investigated the use of Evolutionary Computations (EC) for feature selection [30], feature space transformation [24] and feature construction [16,25,26]. In the context of document recognition, Teredesai and Govindaraju [27,28] have used GP to design active classifiers for off-line script recognition. Their system is based on Park's hierarchical representations [19], with an implicit feature selection. Another relevant work is from Radtke, Wong, and Sabourin [21] who use a multiobjective memetic algorithm to design off-line character representations in a manner similar to ours. But their approach uses genetic algorithms instead of genetic programming.

4.1 Evolution of Representations

Genetic Programming (GP) [3,14] is an EC technique [1, 2] that allows automatic programming of computers by heuristics inspired from natural evolution principles, using genetic operations of crossover and mutation to alter computer programs, and natural selection to choose the fittest programs. Classical GP represents programs as acyclic and undirected graphs (trees), where each node is associated to an elementary operation specific to the problem domain, and where the data type processed and returned by these primitives is usually the same for all nodes. A crossover operation is usually done by exchanging randomly chosen subtrees between two individuals, while a mutation operation consists of replacing a randomly chosen subtree with a new one, also randomly generated.

Classical EC techniques, including GP, typically maximize (or minimize) a single objective function. However, evolutionary multi-objective optimization [4] has emerged in recent years as an important sub-field of EC. Modern population-based multi-objective optimization techniques are based on the concept of Pareto optimality, where solutions are ranked according to a dominance criterion. A solution is said to dominate another if at least one of its objective values is better than the corresponding one for the dominated solution, and all others are at least equal. The Pareto front of a population is defined by the set of non-dominated solutions.

To evolve handwriting character representations, we have implemented a two-objective tree-based GP that merges elements from both the static and data-driven fuzzy regional representations. The data processed by tree nodes consist of two coordinate pairs for the lower-left and upper-right corners of a rectangular region. Table 3 enumerates the building blocks available to the evolutionary process. This set of functional primitives (tree nodes) can be classified into two categories. The first is region-modifying primitives (S2H, S2V, S3H, S3V, S4, S5, D2H, D2V, D3H, D3V, D4, D5, and ZM) that split the current region into sub-regions, or that modify the extent of the current region (ZM only). The second category is feature-extraction primitives (OR, CU, MX, MY, and HW) that extract a given type of features from the current region without modifying its definition. The current region for a given node is always defined by its parent node and the root node simply inherits the full bounding box of the character. There is one additional terminal primitive (T) which has no effect other than closing the tree structure. A weight is associated with each primitive in order to bias its selection probability during initialization and mutation operations. This bias is useful for building an equilibrium between region-modifying and feature-extraction primitives.

Two types of region-modifying primitives are defined. The first type splits the parent region at predetermined width/height fractions (S2H, S2V, S3H, S3V, S4, and

Name	Args	Weight	Description
S2H	2	1.0	Static two region horizontal split.
S2V	2	1.0	Static two region vertical split.
S3H	3	1.0	Static three region horizontal split.
S3V	3	1.0	Static three region vertical split.
S4	4	1.0	Static quad region split.
S5	5	1.0	Static quin region split.
D2H	2	1.0	Data-driven two region horizontal split.
D2V	2	1.0	Data-driven two region vertical split.
D3H	3	1.0	Data-driven three region horizontal split.
D3V	3	1.0	Data-driven three region vertical split.
D4	4	1.0	Data-driven quad region split.
D5	5	1.0	Data-driven quin region split.
ZM	1	1.0	Randomly generated zoom factor.
OR	1	16.0	Randomly generated orientation feature.
CU	1	12.0	Randomly generated curvature feature.
MX	1	4.0	Horizontal center of mass feature.
MY	1	4.0	Vertical center of mass feature.
HW	1	4.0	Height/width ratio feature.
T	0	1.0	Grounding terminal.

Table 3 Primitives used for GP.

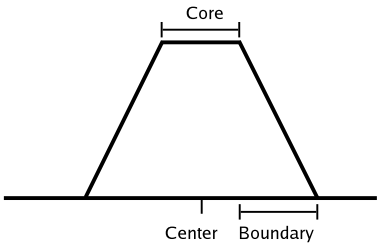

Fig. 6 Fuzzy set shape used to extract orientation and curvature features.

S5), much like it was done in Section 2. It also includes the ZM primitive that changes the scale of the current region by applying a randomly generated zoom factor between 20% and 200%. This zoom value is generated randomly during initialization or mutation, acting as an ephemeral random constant [14] (its value is set to a new random number each time the primitive is mutated). The second type of region modifying primitives (D2H, D2V, D3H, D3V, D4, D5) split the parent region according to the centroid of the strokes found in the parent region, in a way similar to the data-driven representations of Section 3.

Every feature-extraction primitive produces a side-effect by adding a new feature to the output representation without affecting the current region which is simply passed on unchanged to its child node. The feature-extraction primitives are also of two types. The first is composed of orientation and curvature primitives (OR and CU) that extract respectively a degree of orientation and curvature possessed by the strokes contained in the current region. Both primitives include three randomly generated parameters (center, core, and boundary) that specify a symmetric trapezium fuzzy set, as illustrated in Figure 6. As with the zooming primitive, these three parameters are in fact ephemeral random constants generated during initialization or mutation. The ranges of values for the OR primitive are: center in $[-90°, 90°]$, core in $[0°, 30°]$, and boundary in $[5°, 50°]$. Each of these three values is discrete with $5°$ increments in their respective domains. The fuzzy shape for orientation extraction is repeated two times, that is at $-180°$ and $+180°$ from the original center position. For the CU primitive, the center is in $[-0.160, 0.160]$, the core in $[0, 0.160]$ and the boundary in $[0.005, 0.160]$, all three with discrete increments of 0.005. The second type of feature-extraction primitives is composed of the horizontal and vertical center of mass (MX and MY), and the height/width ratio of the region (HW). These primitives do not make use of ephemeral random constants.

4.2 Experimental Results

The experimental protocol used to evolve a population of genetic handwriting representations requires a double cross-validation procedure to avoid overfitting the data during both the evolutionary process and the training of our neural network classifier. The Train-R01/V07 set is thus first randomly decomposed into a fitness evaluation set used to estimate the fitness of individual representations, and a validation set used to retrieve the best-of-run representation. Then, the fitness evaluation set is sub-divided again into a fitness evaluation training set used to update the weights of the MLP classifier, and a fitness evaluation validation set used to halt learning when the neural network starts overfitting the data. In this way, the final validation set is completely independent of the evolutionary process and favors the selection of the individual representation that exhibits the best faculty of generalization; we call this representation the best-of-run representation. Finally, the best-of-run indi-

vidual is re-evaluated on the complete DevTest-R02/V02 set to produce the recognition rates reported below.

This protocol was repeated four times using two different sizes for the validation set: 33% of the whole training set for the first two experiments, 20% for the last two. In each case, the fitness evaluation set (respectively 67% and 80% of the total) was randomly divided into two equal parts: half for the fitness evaluation training set, and half for the fitness evaluation validation set. Moreover, this random sub-division is re-shuffled for each fitness evaluation.

A two-objective optimization is conducted during the evolutionary process. The first objective is to maximize the recognition rate during the training phase. The parameters used for the MLP network are: one hidden layer of 50 neurons, a learning rate of 0.1, a momentum of 0.25, a minimum of 25 training epochs and a maximum of 100, and termination of training after 5 epochs without recognition rate improvements on the fitness evaluation validation set. The second objective is to minimize the total number of features in the representation.

Each evolution ran with populations of 1000 individuals over 100 generations, using a crossover probability of 0.9, mutation (standard, swap, and shrink) probabilities of 0.05 each, and a NSGA-II [5] multi-objective selection operator. Minimum and maximum initial tree depths are set to 3 and 7, respectively, while tree depth is limited to a maximum of 17 levels during evolution. The GP implementation is done in C++ using the Open BEAGLE framework [8,9] and distributed on a 26 nodes Beowulf cluster of 1.2GHz AMD Athlons using Distributed BEAGLE [6,10]. Recognition rates obtained using the best-of-run representations for the four evolutions are given in Table 4. Each best-of-run individual was tested ten times, using the same training methodology as in Sections 2.3 and 3.1.

Results show that with less than half the number of features, the best genetically engineered representation was able to match the recognition rate of the best human designed representation of Section 2.3 (mean rates of 96.40% versus 96.37%, respectively). Moreover, the standard deviations of these mean rates are systematically lower (0.15% versus 0.21% for the above best case). Another interesting observation is that the evolution process is quite stable. Comparable results are obtained for the four distinct evolution runs, even if the larger validation set of the first two (Evol1 and Evol2) appears to have produced representations with fewer features.

Figure 7 summarizes the Evol1 tree structure by showing the region modifying primitives as ellipses, and the feature extraction primitives as rectangular boxes (see Table 3 for a description of the primitives). When several features are extracted within a given region, they are enumerated in a single box without their parameters. For example, "2OR, 1CU" means that two orientation primitives and one curvature primitive were extracted from

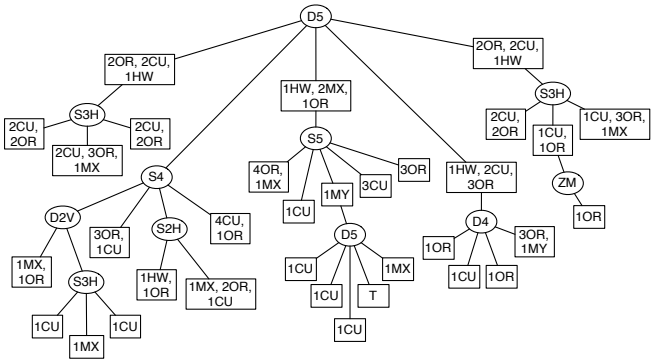

Fig. 7 Tree structure of the Evol1 representation; ellipses represent region modifying primitives while boxes represent feature extraction primitives (see Table 3).

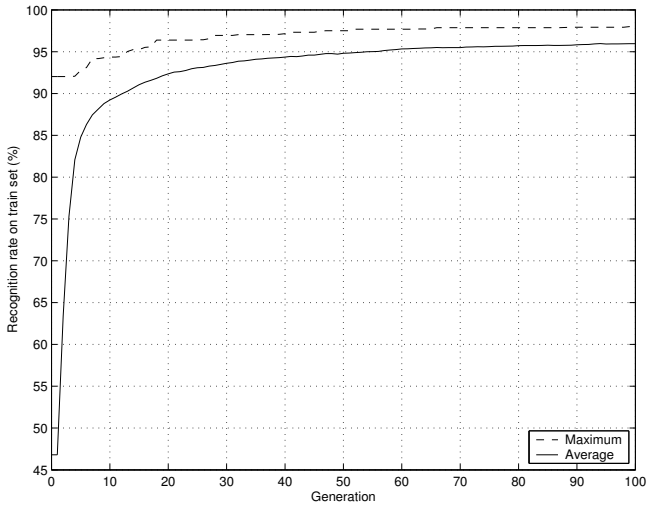

Fig. 8 Evolution of the average and maximum fitness for the Evol1 training data.

the region defined by the parent node. Terminal primitives are omitted unless their parent is region modifying.

This tree structure shows that the evolution process has converged to a mixture of the static and data driven segmentation strategies. It is interesting to note that the highest region modifying node of all four representations is always data driven. So the data driven strategy may not be such a bad idea after all, as long as it is not systematic. For one of the representations (Evol2), the root node is not region modifying. It thus extracts global features. Finally, we can observe that the hierarchical segmentation is mostly 2 and 3-level deep, but sometimes goes up to 4-levels.

For the Evol1 training data, Figure 8 also gives the evolution of the average and maximum individual fitness over the first 100 generations. This graph shows that average convergence is quite rapid. Each evolution took about 2 weeks of execution time on our small cluster. Most of this time is used for training the MLP networks.

Grid Topology	Feature Set Size	Mean Rec. Rate (%)	Max. Rec. Rate (%)	Stdev. Rec. Rate (%)	Mean Shift RFR-3x2	Max. Shift RFR-3x2
Evol1	90	96.40	96.72	0.15	+0.81	+1.12
Evol2	81	96.27	96.57	0.16	+0.67	+0.97
Evol3	112	96.31	96.46	0.13	+0.71	+0.87
Evol4	105	96.37	96.51	0.10	+0.77	+0.91

Table 4 Recognition rate for best-of-run representations obtained from four distinct GP evolutions.

Fig. 9 Characters from DevTest-R02/V02 that were badly classified by at least three out of four classifiers.

In order to gain a better insight into the weaknesses of our evolved character recognizer, we have looked at every badly classified character in the test set. Many of them are badly written or badly segmented, and some are obviously mislabeled. Figure 9 gives a subset of these characters. They are shown in their circular arc stroke reconstruction form, which corresponds to the input of the GP feature extraction module. We assume here that the filtering process that produces this stroke decomposition preserves all of the discriminant information contained in the script, even though we fully recognize that this is a strong hypothesis. The reader should recall that the objective of this paper is to expose a new method for op-

timizing the feature extraction component of a character recognizer, not to report overall best performances.

The characters of Figure 9 are those that were badly classified by at least three out of our four classifiers. They represent about 2.2% of DevTest-R02/V02, section 1a. Examples of incomplete characters are found mostly for digits 0 and 4. For instance, a slash segmented without its associated 0, or the vertical bar of a 4 without its angle part. Some characters are dot like or dash like. Either they are parts of badly segmented characters, or they have been mislabeled and should not even be part of section 1a. Some scripts are completely unrecognizable, like the last instance of characters 0 and 2. Others contain two characters like the first instance of character 7. Others are obviously mislabeled like some instances of characters 2, 3, 4, 6, and 7. Overall, we see that this test set is quite difficult even if some badly classified characters also appear to be recognizable. This may imply that our set of primitives should be augmented with new types of features.

5 Conclusion

We have seen in this paper how counter-intuitive the design of discriminant features can be. For example, contrary to what we expected, the systematic use of the data-driven segmentation introduced in Section 3 has resulted in lower recognition rates than its static counterpart of Section 2. More importantly, we have shown in Section 4 that genetic programming can be used to at least partly automate the trial and error process that most often surround the development of the feature extraction component of pattern recognition systems. This was demonstrated for a particular case of handwriting character recognition, but the general approach is in no way limited to this application.

The proposed approach has the advantage of moving a part of the burden from expensive human development to cheap machine computations, which can be performed nonstop, 24 hours per day. Of course, the problem must be appropriately formulated to allow convergence toward interesting solutions. This was done here by the use of a tree-based hierarchical partitioning of the character bounding box, the use of some high-level feature extraction primitives, and a two-objective process that tries to both minimize the number of features while maximizing the recognition rate. This multi-objective genetic programming can be generalized to many other feature extraction systems, simply by adapting the basic building blocks to the problem at hand, in order to allow extraction of discriminant information.

For the particular case of our character recognizer, even if it did not produce significantly higher recognition rates for our Unipen test data set, the genetic engineering of handwriting representations was able to drastically reduce the feature set size while favoring a more stable classifier training process.

Acknowledgements The authors would like to express their gratitude to A. Schwerdtfeger for proofreading this manuscript.

References

1. Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.): Evolutionary Computation 1: Basic Algorithms and Operators. Institute of Physics Publishing, Bristol, UK (2000)
2. Bäck, T., Hammel, U., Schwefel, H.P.: Evolutionary computation: comments on the history and current state. IEEE transactions on Evolutionary Computation **1**(1), 3–17 (1997)
3. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann, dpunkt.verlag (1998)
4. Coello, C.A.C., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers (2002)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
6. Dubreuil, M., Gagné, C., Parizeau, M.: Distributed BEAGLE: A distributed evolutionary computations extension of Open BEAGLE. http://beagle.gel.ulaval.ca/distributed (2005)
7. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, second edn. John Wiley & Sons, Inc., New York (2001)
8. Gagné, C., Parizeau, M.: Open BEAGLE: A new versatile C++ framework for evolutionary computation. In: Genetic and Evolutionary Computations COnference (GECCO) 2002, Late-Breaking Papers, pp. 161–168. New York, NY, USA (2002)
9. Gagné, C., Parizeau, M.: Open BEAGLE: An evolutionary computation framework in C++. http://beagle.gel.ulaval.ca (2005)
10. Gagné, C., Parizeau, M., Dubreuil, M.: Distributed BEAGLE: An environment for parallel and distributed evolutionary computations. In: Procceedings of the 17th Annual International Symposium on High Performance Computing Systems and Applications (HPCS) 2003, pp. 201–208. Sherbrooke (QC) (2003)
11. Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., Janet, S.: Unipen project of on-line data exchange and recognizer benchmarks. In: Proc. of the 14th Int. Conf. on Pattern Recognition (ICPR), pp. 29–33 (1994)
12. Hagan, M.T., Demuth, H.B., Beale, M.H.: Neural Network Design. PWS Publishing Company (1995)
13. Hébert, J.F., Parizeau, M., Ghazzali, N.: A new fuzzy geometric representation for on-line isolated character recognition. In: Proc. of the 14th International Conference on Pattern Recognition, pp. 33–40 (1998)
14. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
15. Koza, J.R., David Andre, Bennett III, F.H., Keane, M.: Genetic Programming 3: Darwinian Invention and Problem Solving. Morgan Kaufman (1999)
16. Krawiec, K.: Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genetic Programming and Evolvable Machines **3**(4), 329–343 (2002)

17. Li, X., Parizeau, M., Plamondon, R.: Segmentation and reconstruction of on-line handwritten scripts. Pattern Recognition **31**(6), 675–684 (1998)
18. Parizeau, M., Lemieux, A., Gagné, C.: Character recognition experiments using unipen data. In: Proc. of 6th Int. Conf. on Document Analysis and Recognition (ICDAR), pp. 481–485 (2001)
19. Park, J., Govindaraju, V., Srihari, S.N.: OCR in hierarchical feature space. IEEE transactions on Pattern Analysis and Machine Intelligence **22**(4), 400–407 (2000)
20. Plamondon, R., Srihari, S.N.: Online and off-line handwriting recognition: a comprehensive survey. IEEE transactions on Pattern Analysis and Machine Intelligence **22**(1), 63–84 (2000)
21. Radtke, P.V.W., Wong, T., Sabourin, R.: A multi-objective memetic algorithm for intelligent feature extraction. In: Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005). Guanajuato, Mexico (2005)
22. Ratzlaff, E.H.: A scanning n-tuple classifier for online recognition of handwritten digits. In: Proc. of 6th Int. Conf. on Document Analysis and Recognition (ICDAR), pp. 18–22 (2001)
23. Ratzlaff, E.H.: Methods, report and survey for the comparison of diverse isolated character recognition results on the unipen database. In: Proc. of 7th Int. Conf. on Document Analysis and Recognition (ICDAR), pp. 623–628 (2003)
24. Raymer, M.L., Punch, W.F., Goodman, E.D., Kuhn, L.A., Jain, A.K.: Dimensionality reduction using genetic algorithms. IEEE Transactions on Evolutionary Computation **4**(2), 164 (2000)
25. Sherrah, J., Bogner, R.E., Bouzerdoum, A.: Automatic selection of features for classification using genetic programming. In: Proceedings of the 4th Australian and New Zealand Intelligent Information Systems Conference (ANZIIS) 1996 (1996)
26. Sherrah, J., Bogner, R.E., Bouzerdoum, A.: The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic programming. In: Genetic Programming 1997: Proceedings of the Second Annual Conference, pp. 304–312. Morgan Kaufmann, Stanford University, CA, USA (1997)
27. Teredesai, A., Govindaraju, V.: GP-based secondary classifiers. Pattern Recognition **38**(4), 500–512 (2005)
28. Teredesai, A., Park, J., Govindaraju, V.: Active handwritten character recognition using genetic programming. In: Proceedings of EuroGP'01, *LNCS*, vol. 2038, pp. 371–379. Springer-Verlag (2001)
29. Trier, O.D., Jain, A.K., Taxt, T.: Feature extraction methods for character recognition – a survey. Pattern Recognition **29**(4), 641–662 (1996)
30. Yang, J., Honavar, V.: Feature subset selection using A genetic algorithm. IEEE Intelligent Systems **13**, 44–49 (1998)