Generalized vs set median strings for histogram-based distances: algorithms and classification results in the image domain

Christine SOLNON & Jean-Michel JOLION

LIRIS, UMR 5205 CNRS / Université de Lyon 1 / INSA de Lyon
Nautibus, 43 bd du 11 novembre, 69622 Villeurbanne Cedex, France
email: {christine.solnon,jean-michel.jolion}@liris.cnrs.fr

Abstract. We compare different statistical characterizations of a set of strings, for three different histogram-based distances. Given a distance, a set of strings may be characterized by its generalized median, i.e., the string —over the set of all possible strings— that minimizes the sum of distances to every string of the set, or by its set median, i.e., the string of the set that minimizes the sum of distances to every other string of the set. For the first two histogram-based distances, we show that the generalized median string can be computed efficiently; for the third one, which biased histograms with individual substitution costs, we conjecture that this is a NP-hard problem, and we introduce two different heuristic algorithms for approximating it. We experimentally compare the relevance of the three histogram-based distances, and the different statistical characterizations of sets of strings, for classifying images that are represented by strings.

1 Motivations

To manage the huge data sets that are now available, and more particularly classify, recognize or search them, one needs statistical measures to characterize them. This statistical characterization is both well defined and easily computed when data are numerical values, or more generally vectors of numerical values. However, many objects are poorly modelized with such vectors of numerical values, that cannot express the sequentiality of attributes. Strings are symbolic structures that allow a richer modelization by integrating a notion of order.

To exploit sets of strings, one needs a statistical characterization of these sets. This characterization depends on a distance measure, that quantifies the dissimilarity of two strings: given a distance, a set of strings may be characterized by its generalized median, i.e., the string —over the set of all possible strings— that minimizes the sum of distances to every string of the set, or by its set median, i.e., the string of the set that minimizes the sum of distances to every other string of the set.

The complexity of the computation of generalized and set median strings depends on the considered distance. For example, for the well known edit distance of Levenshtein, the set median string may be computed in polynomial time, whereas the computation of the generalized median string is a NP-hard problem [dlHC00,SP01].

In this paper, we focus on three histogram-based distances for strings: the first one, called d_H, considers strings as sets of symbols and is basically defined as a sum of differences of distributions of symbols; the second one, called d_{H_ω} integrates a notion of order by associating weights to positions in strings; the third one, called $d_{H_{\omega,c}}$, biased distances with individual substitution costs of symbols occurring at a same position, in order to express the fact that some symbols are rather similar, whereas some others are very different. These three histogram-based distances have the same computational complexity, which is linear with respect to the size of the strings and the alphabet, and are an order quicker than the edit distance.

A goal of this paper is to study statistical characterizations of sets of strings when considering these histogram-based distances. For the first two distances, we show that the generalized median string can be computed efficiently; for the third one, that biased histograms with individual substitution costs, we conjecture that this is a NP-hard problem, and we introduce two different heuristic algorithms for approximating it.

An application in image classification is proposed as an illustration of these results.

2 Background

2.1 Notations

The alphabet is noted $\mathcal{A}$ and symbols of $\mathcal{A}$ are noted α_i with $1 \leq i \leq |\mathcal{A}|$. Strings are finite length sequences of symbols from $\mathcal{A}$ and are noted s_j with $j \geq 1$. The set of all strings from $\mathcal{A}$ is noted $\mathcal{A}^*$. The length of a string s_j is noted $|s_j|$, and the k^{th} symbol of a string s_j is noted s_j^k.

2.2 Statistical characterisation of a set of strings

Let $d : \mathcal{A}^* \times \mathcal{A}^* \rightarrow \mathbb{R}^+$ be a distance or a dissimilarity measure for any pair of strings from $\mathcal{A}$ (see 3). The first moment, also called *generalized median*, of a set of strings $S \subseteq \mathcal{A}^*$ is defined as a string of $\mathcal{A}^*$ that minimizes the sum of distances to every string of S, i.e.,

$$\text{generalized\_median}(S) = arg \min_{s_{j_1} \in \mathcal{A}^*} \sum_{s_{j_2} \in S} d(s_{j_1}, s_{j_2}) \tag{1}$$

The complexity of the computation of the generalized median string depends on the distance considered. When this complexity is too high, one may approximate the generalized median string by constraining the search to the set S, yielding the *set median* of S as

$$\text{set\_median}(S) = arg \min_{s_{j_1} \in S} \sum_{s_{j_2} \in S} d(s_{j_1}, s_{j_2}) \tag{2}$$

3 Distances between strings

3.1 Edit distance

The most famous distance between strings has been proposed by Levenshtein, e.g., the edit distance [Lev66]. The edit distance between two strings s_{j_1} and s_{j_2}, denoted by $d_e(s_{j_1}, s_{j_2})$, is defined by the minimum cost set of edit operations required to transform s_{j_1} into s_{j_2}. Three edit operations are allowed (substitution of a symbol by another symbol, deletion of a symbol, and insertion of a symbol); costs are associated with these operations. A simple algorithm using dynamic programming for computing the edit distance can be found in [WF74]. Its computational time complexity is in $\mathcal{O}(|s_{j_1}| \cdot |s_{j_2}|)$.

For this edit distance, the computation of the generalized median string is a NP-hard problem [dlHC00,SP01]. The generalized median string may be approximated by using heuristic algorithms, such as greedy algorithms [MHJC00] or genetic search [JBC04].

3.2 Histogram-based distance d_H

An alternative to the edit distance is to consider a sequence not as a string but as a set of symbols. Thus a basic distance between two sets is the comparison of the distributions of symbols defined as:

$$d_H(s_{j_1}, s_{j_2}) = \sum_{\alpha_i \in \mathcal{A}} abs(H(s_{j_1}, \alpha_i) - H(s_{j_2}, \alpha_i)) \tag{3}$$

where $H(s_j, \alpha_i)$ is the number of occurrences of symbol α_i in string s_j, and abs is the function that returns the absolute value[1]. The main advantage of this distance is its computational cost which is in $\mathcal{O}(|s_{j_1}| + |s_{j_2}| + |\mathcal{A}|)$[2]. For strings of different sizes, the histograms must be normalized before comparison.

For this histogram-based distance, the generalized median string of a set of strings S can be constructed as follows: starting from an empty string, for each symbol $\alpha_i \in \mathcal{A}$, insert k times the symbol α_i to the string, where k is the median value of the set $\{H(s_j, \alpha_i), s_j \in S\}$. This generalized median string can be computed in $\mathcal{O}(|S| \cdot (l + |\mathcal{A}|))$, where l is the length of the strings of S[3].

[1] There exists many other different histogram-based distances such as, e.g., kullback-Leibler or Kolmogorov-Smirnov. Our work, based on a distance defined by means of absolute differences of distributions, could be extended to other histogram-based distances as well.

[2] In case of very large alphabets, one may use a hashing table in order to consider only the symbols of the alphabet that actually occur in the strings, thus computing the distance in $\mathcal{O}(|s_{j_1}| + |s_{j_2}|)$.

[3] Note that the median element of a set can be selected in linear time with respect to the size of the set by using a "divide-and-conquer" approach similar to the one used for the quicksort [CLR90]: the idea is to partition the set in two parts containing elements greater than (resp. lower or equal to) a given element; depending on the cardinalities of these two parts, the search for the median element can be recursively continued in one of the two parts.

3.3 Weighted histogram-based distance d_{H_ω}

The histogram-based distance d_H does not take into account the order the symbols appear in the strings. One could integrate information on the sequentiality of the symbols by using n-grams, thus comparing the distributions of sub-sequences of n symbols. However, in some applications (as the one described in section 5), the order of the symbols in a string may not express a strong sequentiality, but a difference in the importance of the symbols. In this case, the fact that a symbol is just before another symbol is not very significant; the main information contained in the string structuring is the global position of symbols, those at the beginning of a string being more important than those at the end.

In this case of decreasing importance strings, one may associate a weight ω_k with every position k in strings. To emphasize differences at the beginning of the strings, this weight may be defined, e.g., by $\omega_k = 1 + l - k$ where l is the length of the string. To compare strings of different sizes, it is then necessary to complete the shortest string with a new extra symbol until the two strings have the same size.

Hence, we define the weighted histogram associated with a string s_j and a symbol α_i:

$$H_\omega(s_j, \alpha_i) = \sum_{1 \le k \le |s_j|, s_j^k = \alpha_i} \omega_k$$

and the weighted histogram-based distance between two strings s_{j_1} and s_{j_2}:

$$d_{H_\omega}(s_{j_1}, s_{j_2}) = \sum_{\alpha_i \in \mathcal{A}} abs(H_\omega(s_{j_1}, \alpha_i) - H_\omega(s_{j_2}, \alpha_i)) \qquad (4)$$

This distance has the same computational cost than the basic histogram-based distance d_H.

For this weighted histogram-based distance, one can construct a "generalized median weighted histogram" of a set of strings S as follows: for each symbol $\alpha_i \in \mathcal{A}$, set the weighted histogram value associated with α_i to the median value of the set $\{H_\omega(s_j, \alpha_i), s_j \in S\}$. This generalized median weighted histogram can be computed within the same time complexity than for the histogram-based distance, i.e., in $\mathcal{O}(|S| \cdot (l + |\mathcal{A}|))$, where l is the length of the strings of S. Note that it may not be possible to construct a string corresponding to this weighted histogram. However, it may be used to statistically characterize a set of strings.

3.4 Weighted histogram-based distance with substitution costs $d_{H_{\omega,c}}$

The histogram-based distances d_H and d_{H_ω} assume that all symbols are "equally different". However, some symbols may be considered as rather similar, whereas some others may be very different. Therefore, [RLJS05] has proposed a new distance, which has the same computational complexity as d_H and d_{H_ω}, but which is biased with individual substitution costs of symbols occurring at a same position.

This new distance is based on weighted histograms with substitution costs. Given two strings s_{j_1} and s_{j_2} and a symbol α_i, these histograms are defined as follows

$$H_{\omega,c}(s_{j_1}, \alpha_i) = \sum_{1 \le k \le |s_{j_1}|, s_{j_1}^k = \alpha_i} \omega_k \cdot c(\alpha_i, s_{j_2}^k)$$

$$H_{\omega,c}(s_{j_2}, \alpha_i) = \sum_{1 \le k \le |s_{j_2}|, s_{j_2}^k = \alpha_i} \omega_k \cdot c(s_{j_1}^k, \alpha_i)$$

where $c : \mathcal{A} \times \mathcal{A} \to \mathbb{R}^+$ is a function which defines the cost of substituting one symbol by another symbol.

Then, the weighted histogram-based distance with substitution costs is defined by

$$d_{H_{\omega,c}}(s_{j_1}, s_{j_2}) = \sum_{\alpha_i \in \mathcal{A}} abs(H_{\omega,c}(s_{j_1}, \alpha_i) - H_{\omega,c}(s_{j_2}, \alpha_i)) \tag{5}$$

Note that this distance is not a metric, and does not satisfy the triangular inequality property.

The fact that the histogram is biased by the individual substitution cost of every pair of symbols occurring at a same position implies that one cannot construct a "generalized median weighted histogram" of a set of strings S, independently from any candidate median string. Therefore, we conjecture that the computation of the generalized median string of a set of strings is NP-hard.

4 Approximations of the generalized median string for $d_{H_{\omega,c}}$

This section describes two algorithms for approximating the generalized median string of a set of strings $S \subseteq \mathcal{A}^*$, when considering the weighted histogram-based distance with substitution costs $d_{H_{\omega,c}}$.

We shall assume that all strings of S have the same length l: if this is not the case, it is always possible to complete every string that is shorter than l with a new extra symbol.

4.1 Greedy algorithm

The generalized median string of S may be approximated in a greedy way: starting from an empty string s_{greedy}, symbols are iteratively added at the end of s_{greedy} until the length of s_{greedy} is equal to l. At each step, one selects the symbol $\alpha_i \in \mathcal{A}$ that minimizes the sum of distances between $s_{greedy} \cdot \alpha_i$ and every string of S (restricted to the $|s_{greedy}| + 1$ first symbols).

A key point to keep a low time complexity is to incrementally evaluate the sum of distances induced by each candidate symbol. This is done by maintaining, at each iteration $l' \le l$:

– for every string $s_j \in S$, two arrays H_1^j and H_2^j such that for every symbol $\alpha_i \in \mathcal{A}$:

$$H_1^j[\alpha_i] = \sum_{1 \leq k < l', s_j^k = \alpha_i} \omega_k \cdot c(\alpha_i, s_{greedy}^k)$$

$$H_2^j[\alpha_i] = \sum_{1 \leq k < l', s_{greedy}^k = \alpha_i} \omega_k \cdot c(\alpha_i, s_j^k)$$

– an array *sum* such that for every string $s_j \in S$,

$$sum[s_j] = \sum_{\alpha_i \in \mathcal{A}} abs(H_1^j[\alpha_i] - H_2^j[\alpha_i])$$

Thanks to these data structures, the choice of the next symbol to add is done in $\mathcal{O}(|\mathcal{A}| \cdot |S|)$. Each time a new symbol is added at the end of the string, these data structures are updated in $\mathcal{O}(|S|)$. As a consequence, the time complexity of the greedy algorithm is in $\mathcal{O}(l \cdot |\mathcal{A}| \cdot |S|)$.

4.2 Local search

The generalized median string of S may also be approximated by iteratively modifying an initial string of length l: at each iteration, a symbol of the string is replaced by a new symbol such that the sum of distances to every string of S is decreased; these replacements are performed until no more replacement can decrease the sum of distances, thus obtaining a locally optimal string that cannot be improved by a simple replacement.

We have compared different strategies (including meta-heuristics such as tabu search) for selecting the next replacement to perform at each step. On average, the best compromise between solution quality and CPU-time has been reached when considering a "first-improvement" strategy, i.e., when selecting the first found replacement that decreases the sum of distances.

This local search process may be started from different initial strings, e.g., from the set median string of S, from the string constructed by the greedy algorithm, or a string which is randomly generated from $\mathcal{A}^*$.

The same data structures than for the greedy algorithm may be used to evaluate replacements at low cost. With such data structures, given a position k and a new symbol α_i, the replacement of the symbol at position k by α_i may be done in $\mathcal{O}(|S|)$ so that the time complexity of the local search algorithm is in $\mathcal{O}(n \cdot |S|)$ where n is the number of replacements that are evaluated. Of course, n depends on the strategies considered for selecting the replacements to perform and for building the initial string from which starting the local search; it also depends on the length of the string.

4.3 Experimental results

Table 1 compares the quality of the different approximations of the generalized median string introduced previously, e.g., the set median string, the string

Length	set median	greedy	LS(set median)	LS(greedy)	LS(random)
100	468.87	13.76%	13.63%	14.28%	**14.31%**
200	425.57	16.56%	16.50%	**17.20%**	16.98%
400	381.29	19.41%	19.18%	**20.06%**	19.79%
800	329.93	21.29%	20.93%	**22.09%**	21.63%

Table 1. Comparison of approximations of the generalized median string: each line first gives the length of the strings and the sum of distances to the set median string, and then the percentage of improvement of this sum of distances when considering strings computed by the greedy and local search algorithms (average results for the 10 classes of the SIMPLIcity base described in 5, each class having 100 strings).

Length	set median	greedy	LS(set median)	LS(greedy)	LS(random)
100	0.02	0.29	2.59	1.67	2.32
200	0.03	0.51	5.44	4.40	5.03
400	0.04	1.01	14.62	10.38	13.35
800	0.09	2.13	33.45	32.24	34.47

Table 2. Comparison of CPU-times: each line displays the length of the strings and the CPU times (in seconds) spent to compute the different approximate generalized median strings (average results for the 10 classes of the SIMPLIcity base described in 5, each class having 100 strings).

computed by the greedy algorithm (greedy), and the strings computed by the local search algorithm starting from different initial strings, i.e., from the set median string (LS(set median)), the string computed by the greedy algorithm (LS(greedy)) and a randomly generated string (LS(random)).

This comparison is done on strings of the SIMPLIcity base described in section 5. This base contains 10 classes of 100 strings of 4000 symbols. The table gives average results on the 10 classes, when successively limiting the length of the strings to the 100, 200, 400, and 800 first symbols. For each length, the table first displays the sum of distances to the set median string, and then, for each approximation of the generalized median string, the percentage of improvement with respect to this sum of distances.

Both greedy and local search algorithms significantly better approximate the generalized median string than the set median string. The best improvements are usually obtained by local search, when it is started from the string generated by the greedy algorithm. Surprisingly, starting local search from the set median string often leads to a slightly worse approximation than starting from a randomly generated string. However, all approximations obtain rather close results.

Table 1 also shows that the larger the strings, the better improvements: when strings are limited to the 100 first symbols, the sum of distances to the greedy approximation is 13.76% as small as the sum of distances to the set median string; when considering the 200 (resp. 400 and 800) first symbols, this percentage of improvement rises to 16.56 (resp. 19.41 and 21.29).

Table 2 compares CPU-times spent to compute the different approximations on a 2.16GHz Intel dual core with a 2MB cache. This table shows us that computing the set median string is more than ten times as fast as computing an approximation with the greedy algorithm, which itself is more than ten times as fast as computing an approximation with the local search approaches. Also, when starting local search from the string generated by the greedy algorithm, CPU time is slightly smaller than when starting from a set median or a randomly generated string.

The quality improvement is thus balanced by the CPU-time cost. However, in applications such as classification of unknown strings in already known clusters, the best representative of each cluster, e.g., the generalized median string, is computed off-line, one-for-all.

5 Classification results in the image domain

5.1 Representing images by strings

We introduced in [SJ05] a new representation of images based on strings of symbols. This signature, both precise and compact, is based on notions such as interest points, contrast and order. First, a given image is binarized such that it keeps all the contrasts. Then local maxima of the contrast energy are extracted and associated with their local 3×3 binary neighborhood in the binary image. We thus get a 2D map of symbols, e.g. the 3×3 binary patterns. As any local maxima, e.g. interest points, is also characterized by a measure of contrast energy, we use this measure to sort the points, yielding a string of symbols. The contrast energy measure is no longer kept in the final signature. In this application, the alphabet is made of 512 symbols, corresponding to the 2^9 different 3×3 binary neighborhoods.

Note that with this representation of images by strings, the edit distance of Levenshtein is not relevant and gives very disappointing results for classification purposes. Indeed, the edit distance mainly considers the “local” order of symbols —their relative positions— whereas we are more interested in a “global” order of symbols —their global positions in the string, as symbols are sorted with respect to their contrast energy and we consider very long strings: we mainly want to distinguish symbols with high contrast energy, at the beginning of the strings, from symbols with low contrast energy, at the end of the strings. Moreover, the edit distance is an order slowler, which makes it prohibitive on large strings of more than one thousand symbols length.

5.2 Test suite and experimental settings

We have performed experiments on the SIMPLIcity database [WLW01] which contains 1000 images of size 384×256 extracted from the well known old commercial COREL database[4]. The database contains ten clusters representing semantic generalized medianingful categories such as Africa people and villages, beaches, buildings, buses, dinosaurs, elephants, flowers, food, forses and mountains and glaciers. There are 100 images per cluster. Each image of the database is represented by a string of 4000 symbols max, as described in the previous section.

For distances d_{H_ω} and $d_{H_{\omega,c}}$, which associate a weight ω_k with every position k in strings, we have defined $\omega_k = l - k + 1$, where l is the length of the strings, in order to emphasize differences at the beginning of the strings.

For the distance $d_{H_{\omega,c}}$, which biased histograms with individual substitution costs, we have tuned costs for this database by a basic adaptive process. Let M be a 512×512 matrix initialized to 0. We scan all the possible pairs of symbols $(s^k_{j_1}, s^k_{j_2})$. If the strings s_{j_1} and s_{j_2} belong to the same cluster, $M(s^k_{j_1}, s^k_{j_2})$ is decreased by 1 else it is increased by $\frac{1}{1-NC}$ where NC is the number of clusters (in order to take into account the *a priori* probability of two strings to belong to the same cluster). The final cost matrix is then discretized based on the sign of M and we set each cost $c(\alpha_{i_1}, \alpha_{i_2})$ to 1 (resp. 2 and 3) if $M(\alpha_{i_1}, \alpha_{i_2})$ is negative (resp. null and positive).

We have classified the strings extracted from the SIMPLIcity base according to a nearest neighbour approach: to classify a string, we compute the distance between this string and the representative of every class (the set median, or an exact or approximated generalized median); the closest representative determines the class. We have computed representatives of every class according to a “leave-out-one” principle: the string which is classified is removed from its class before computing its representatives.

We have performed experiments with different lengths of strings: strings extracted from images have 4000 symbols (some strings were shorter, but we have completed them with a new extra symbol); to study the influence of the length of the strings, we report experimental results obtained when limiting the number of symbols to different lengths varying from 50 to 2500.

5.3 Experimental results

We now compare the different histogram-based distances (d_H, d_{H_ω}, and $d_{H_{\omega,c}}$), and the different statistical characterizations (set median string, exact generalized median string for d_H and d_{H_ω}, and approximated generalized median string for $d_{H_{\omega,c}}$), for classifying strings representing images of the SIMPLIcity database.

Table 3 compares global classification rates (GCR), i.e., percentages of strings which have been assigned to the right classes. Let us first compare GCR when classes are characterized by set median strings for the three different histogram-based distances introduced in 3. We note that introducing weights ω_k to emphasize differences at the beginning of the strings improves GCR when strings are

[4] The SIMPLIcity database can be downloaded on the James Z. Wang web site at http://wang.ist.psu.edu/jwang/test1.tar.

	set median strings			(exact or approximated) generalized median strings			
Length	H	H_ω	$H_{\omega,c}$	H	H_ω	$H_{\omega,c}$	
						Greedy	$LS(Greedy)$
50	28.4	27.2	35.6	33.2 (+4.8)	33.4 (+6.2)	41.0 (+5.4)	**43.9** (+8.3)
100	35.2	34.3	41.4	43.7 (+8.5)	41.0 (+6.7)	**45.7** (+4.3)	44.6 (+3.2)
300	44.1	45.3	48.6	**60.5** (+16.4)	57.8 (+12.5)	55.0 (+6.4)	56.8 (+8.2)
500	55.3	48.1	52.5	**63.8** (+8.5)	61.8 (+13.7)	61.0 (+8.5)	61.7 (+9.2)
800	57.2	52.5	61.9	**68.1** (+10.9)	66.4 (+13.9)	65.1 (+3.2)	63.9 (+2.0)
1000	59.5	57.8	60.8	**69.6** (+10.1)	67.8 (+10.0)	65.1 (+4.3)	65.3 (+4.5)
1250	62.0	58.4	63.4	**69.7** (+7.7)	68.9 (+10.5)	67.4 (+4.0)	66.8 (+3.4)
1500	60.7	61.9	65.5	70.3 (+9.6)	**70.5** (+8.6)	68.6 (+3.1)	69.4 (+3.9)
1750	62.9	63.3	63.9	68.8 (+5.9)	**71.8** (+8.5)	67.9 (+4.0)	68.7 (+4.8)
2000	57.5	64.3	62.7	63.9 (+6.4)	**71.4** (+7.1)	68.2 (+5.5)	68.0 (+5.3)
2500	53.7	61.0	62.6	63.3 (+9.6)	**70.1** (+9.1)	66.5 (+3.9)	65.9 (+3.3)
avg.	52.4	52.2	56.3	61.4 (+9.0)	61.9 (+9.7)	61.0 (+4.7)	61.4 (+5.1)

Table 3. Comparison of global classification rates (GCR) (average results for the 10 classes of the SIMPLIcity base, each class having 100 strings). Each line successively displays the length of the strings, the GCR obtained when representatives are set median strings (for distances d_H, d_{H_ω}, and $d_{H_{\omega,c}}$), and the GCR obtained when representatives are generalized median strings (for distances d_H and d_{H_ω}), and approximations computed by greedy and local search algorithms (for distance $d_{H_{\omega,c}}$); GCR obtained with (exact or approximated) generalized median strings are followed in brackets by the improvement with respect to the set median string.

long enough (i.e., for lengths greater than a thousand or so symbols), whereas it decreases GCR for shorter strings. Note also that introducing individual substitution costs significantly improves GCR.

Let us now compare GCR when classes are characterized by exact or approximated generalized median strings. We note that these (approximated) generalized median strings are better representatives than set median strings. However, exact generalized median strings, computed for the distances d_H and d_{H_ω}, improve more significantly GCR than approximated ones, computed for the distance $d_{H_{\omega,c}}$: on average, the GCR is improved by 9 (resp. 9.7) points for d_H (resp. d_{H_ω}) when representing classes by generalized instead of set median strings; however, this GCR is only improved by 4.7 (resp. 5.1) points for $d_{H_{\omega,c}}$ when representing classes by approximations computed by the greedy (resp. local search) algorithm.

Note finally that GCR obtained with approximations computed by local search are not significantly better than GCR obtained with approximations computed by the greedy algorithm: on average, the GCR is improved of 0.4 points only.

6 Discussion

We introduced in this paper three histogram-based distances for strings. For the first two ones, the generalized median string can be computed in polynomial time, and we experimentally show that classification is significantly improved when characterizing classes with generalized median strings instead of set median strings. However, we conjecture that the computation of generalized median strings for the third histogram-based distance is a NP-hard problem, so that we have proposed two heuristic algorithms for approximating generalized median strings in this case. Experimental results showed us that, if classification is improved when characterizing classes with these approximations, they are not as relevant as we would like and improvements are twice as small as improvements obtained with exact generalized median strings. Hence, further work will first concern an explanation of these disappointing results: are they due to the fact that our heuristic algorithms build approximations that are far from optimality, or are they due to the distance itself? Actually, we need information on the distribution of strings with respect to distances. We thus are currently working on the definition of a probability density function on such space and algorithms to approximate this function.

Another trend will be to relate our string-based approach to the more usual graph-based representation of images. Of course, we shall investigate the seriation of a graph-based representation of an image but also alternatives such as graphs or trees of strings, each string being related to a localized area in an image.

References

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[dlHC00] C. de la Higuera and F. Casacuberta. Topology of strings: Median string is np-complete. *Theoretical Computer Science*, 230(1/2):39–48, 2000.

[JBC04] X. Jiang, H. Bunke, and J. Csirik. Median strings: A review. In M. Last, A. Kandel, and H. Bunke (eds) World Scientific, editors, *Data Mining in Time Series Databases*, pages 173–192, 2004.

[Lev66] A. Levenstein. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phy. Dohl.*, 10:707–710, 1966.

[MHJC00] C.D. Martinez-Hinarejos, A. Juan, and F. Casacuberta. Use of median string for classification. In *International Conference on Pattern Recognition*, volume 2, pages 903–906, 2000.

[RLJS05] J. Ros, C. Laurent, J.M. Jolion, and I. Simand. Comparing string representations and distances in a natural image classification task. In LNCS Springer, editor, *4th IAPR International Workshop on Graph based Representations*, volume 3434, pages 71–83, 2005.

[SJ05] I. Simand and J.M. Jolion. Représentation d'images par chaînes de symboles: application à la recherche par le contenu. In Presses universitaires de Louvain, editor, *Actes du 20ème colloque GRETSI: Traitement du signal et des images*, volume 2, pages 925–928, 2005.

[SP01] J.S. Sim and K. Park. The consensus string problem for a metric is np-complete. *Journal of Discrete Algorithms*, 2(1):115–121, 2001.

[WF74] R.A. Wagner and M.J. Fisher. The string to string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[WLW01] J.Z. Wang, J. Li, and G. Wiederhold. Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.