A Mean String Algorithm to Compute the Average Among a Set of 2D Shapes

Gemma Sánchez [a,b] Josep Lladós [a] Karl Tombre [b]

[a] *Computer Vision Center, Dept. Informàtica. Universitat Autònoma de Barcelona, 08193 Bellaterra (Barcelona), Spain*

[b] *Loria, Campus scientifique, B.P. 239, 54506 Vandœuvre-lès-Nancy CEDEX, France*

Abstract

An algorithm to compute the mean shape, when the shape is represented by a string, is presented as a modification of the well-known string edit algorithm. Given N strings of symbols, a string edit sequence defines a mapping between their corresponding symbols. We transform these sets of mapped symbols (edges) into piecewise linear functions and we compute their mean. To transform them into functions, we use the equation of the line defining their edges, and the percentage of their length, in order to have a common parameterization. The algorithm has been experimentally tested in the computation of a representative among a class of shapes in a clustering procedure in the domain of a graphics recognition application.

Key words: String edit distance, Mean string, Mean shape, 2D Shapes, Median String

Corresponding author:
Gemma Sánchez
Centre de Visió per Computador
Edifici O
Universitat Autònoma de Barcelona
08193 – Bellaterra
SPAIN
Tlf: +34 93 581 27 80
Fax: +34 93 581 16 70
e-mail:gemma@cvc.uab.es

1 Introduction

Structural and syntactic pattern recognition methods often need reference models for a set of shapes, e.g. to compute some distance between a shape to be recognized and the available models. Given a distance, such a representative shape should be considered as some kind of mean shape, i.e. a shape whose characteristics have the mean values of the characteristics of the different shapes in the family.

To compute such a reference shape, a first possible approach consists in finding the *median shape*, i.e. to choose the representative among the shapes in the set. Another approach, which we will discuss in this paper, consists in computing a *mean shape*, which does not necessarily belong to the set of shapes. But in order to compute such a mean shape, we need to solve an object prototype learning problem or to be able to infer a structure from the clustering analysis of regular structural clusters. In the first case, a model can be inferred by computing the mean shape of a set of noisy samples of the same model. In the second case, the mean shape of all the shapes in a cluster will be the representative of the cluster.

Various representations and techniques have been proposed to solve the problem of computing the mean shape. Valveny [14] uses deformable models. Cootes et al. [7] use Point Distribution Models. Ueda and Suzuki [13] use multiscale convex/concave structure. Jiang et al. [8], Bunke et al. [3] and Bunke and Kandel [2] use directed labeled graphs.

A number of methods have been proposed to compute a mean string from a set of strings [4,6,9,10]. We know that a 2D shape can be represented by a cyclic string. Although mean string computations are not directly applicable to the computation of a mean shape, they provide useful hints on the way a mean shape can be computed, when it is represented by a string.

There are basically two problems in adapting the computation of the general mean string to a string that represents shapes. First, we have to select the starting symbol of the string, which is not uniquely defined when the contour is closed—which means that the string is cyclic, to the contrary of usual strings of characters. The second problem is that a string is usually formed by symbols that belong to a finite set of symbols, whereas this set of "symbols" may become infinitely large when we want to represent shapes with good accuracy.

In this paper, we present a method to compute the mean shape, based on the well-known string edit distance algorithm. Shapes are represented by cyclic strings, so that the mean shape is in fact the mean string. Hence, we will use from now both terms "mean shape" and "mean string" to refer to the same concept. The modifications that we propose in this well-known algorithm

allow us to compute the mean string as well. With the string edit distance, we have the correspondences between edges or edge sets of both strings. From these correspondences, we compute the mean edges which constitute the mean shape, treating them as piecewise linear functions and computing the mean shape as a mean between functions. The transformation from a set of edges to a piecewise linear function is determined using the relative length of the edges as a parameter.

The organization of this paper is as follows. In Section 2, we introduce the string concept and the well-known string edit distance, and we formally define the concept of mean string. In Section 3 we explain our algorithm. In Section 5, we show and discuss some results using the algorithm. Finally, Section 6 is devoted to conclusions.

2 Formulation of the Mean String

The formulation of the mean string is based on the string edit distance algorithm. Let us first introduce some basic definitions and notations from string theory.

Let Σ be an alphabet of symbols, Σ^* the set of all finite strings over Σ and $A = a_1 \dots a_n \in \Sigma^*$, $B = b_1 \dots b_m \in \Sigma^*$ two strings; $n, m \geq 0$. The distance between A and B, $d(A, B)$, is defined in terms of elementary edit operations required to transform A into B with minimum cost. Conventionally, three edit operations are defined:

- *substitution* of a symbol $a \in \Sigma$ in A by a symbol $b \in \Sigma$ in B, denoted as $a \rightarrow b$.
- *insertion* of a symbol $b \in \Sigma$ in B, denoted as $\lambda \rightarrow b$.
- *deletion* of a symbol $a \in \Sigma$ in A, denoted as $a \rightarrow \lambda$.

where λ denotes the empty string. The well-known algorithm of Wagner and Fischer [15] computes $d(A, B)$ in $O(nm)$ time and space.

2D shapes can be represented by cyclic strings [19]. To keep the cost polynomial, Bunke and Bühler [1] proposed a sub-optimal method where they formulated the *cyclic string edit distance* $d_c(A, B)$ in terms of the classical string edit distance between A and B^2 where B^2 is the string BB. Other methods have been recently proposed to achieve better computation times [20].

Let A be a string $A = a_1 a_2 \dots a_n$. A string $B = b_1 \dots b_m$, $m \leq n$, is a *substring* of A if $A = a_1 \dots a_{i-1} B a_{j+1} \dots a_n$. The substring B will be denoted as $A_{i,j} = a_i \dots a_j$, $1 \leq i, j \leq n$. Let $A \in \Sigma^*$; the *merging operation*, introduced by Tsai

and Yu [11], is denoted as $A_{i,j} \rightarrow a$ and defined as the edit operation that transforms the substring $A_{i,j}$ into a symbol $a \in \Sigma$. The symbol a is defined as an approximation of the substring $A_{i,j}$, in our case such definition is introduced in Section 3. Equivalent to the three basic edit operations, the merge operation will involve a *merging cost* function denoted as $c(A_{i,j} \rightarrow a)$. This operation allows us to compute a *block substitution* [1], i.e. the substitution of a whole sequence of symbols by another sequence of symbols. A *block substitution* is denoted as $A_{i,j} \rightarrow B_{k,l}$. To perform this substitution, we merge $A_{i,j}$ into a symbol a and the substring $B_{k,l}$ into a symbol b. The block substitution has an associated cost that is defined as $c(A_{i,j} \rightarrow a) + c(B_{k,l} \rightarrow b) + c(a \rightarrow b)$.

One of the early attempts to solve the mean string problem was proposed by Kruskal [9]; later, Lopresti and Zhou [10] used a similar algorithm to compute the mean of several OCRed strings. Our algorithm is inspired by these two previous works, but we work with strings that represent 2D shapes, so the geometrical information they convey must be taken into account.

We need to distinguish two concepts: the *set median string* and the *mean string*, also called *generalized median string*.

Definition 1 *The* set median string SM *of a set of strings* $S = \{S^1, \ldots, S^N\}$ *is defined in terms of the edit distance, as the string sequence in the set* S *that minimizes the combined cost of editing* SM *into each of the* S^i*. It follows that* $SD_c(S^1, \ldots, S^N)$ *represents the* minimum edit cost *to transform* SM *into each of the* S^i*, defined as:*

$$SD_c(S^1, \ldots, S^N) \equiv \min_{SM \in S} \sum_{i=1}^{N} d_c(SM, S^i) \tag{1}$$

where $d_c(SM, S^i)$ *represents the cyclic edit distance between* SM *and* S^i*.*

Definition 2 *The* mean string M *or the* generalized median string *of a set of strings* $S = \{S^1, S^2, \ldots, S^N\}$ *is defined in terms of the edit distance, as the sequence that minimizes the combined cost of editing* M *into each of the* S^i*. Representing* $D_c(S^1, \ldots, S^N)$ *the* minimum edit cost *to transform* M *into each of the* S^i*, the definition of* M *is:*

$$D_c(S^1, \ldots, S^N) \equiv \min_{M \in \sum^*} \sum_{i=1}^{N} d_c(M, S^i) \tag{2}$$

where $d_c(M, S^i)$ *represents the cyclic edit distance between* M *and each* S^i*.*

In general M is not unique and the *set of all possible mean strings* is defined as:

Definition 3 *Let* $S = \{S^1, \ldots, S^N\}$ *be a set of strings, the* set of all possible

mean strings *is*

$$Mean(S^1,\ldots,S^N) \equiv \{M \in \Sigma^* | \sum_{i=1}^{N} d_c(M,S^i) = D_c(S^1,\ldots,S^N)\} \qquad (3)$$

where we are interested in finding only one representative mean string $M \in Mean(S^1,\ldots,S^N)$.

As proposed in previous approaches [4,10], the mean string inference can be performed by introducing the computation of D_c in the cyclic string matching algorithm described in this Section. To describe the algorithm for computing D_c, we need to define δ as the cost to transform a mean substring of a set of substrings into them:

$$\delta(S^1_{b_1,e_1},\ldots,S^N_{b_N,e_N}) = min_{X\in\Sigma^*} \sum_{j=1}^{N} c(X \rightarrow S^j_{b_j,e_j}) \qquad (4)$$

The algorithm for computing D_c—and hence M—involves the construction of a multi-dimensional distance table T_D, while M is being constructed. The recurrence for N strings $S^1 = s^1_1 s^1_2 \ldots s^1_{l_1}$, ..., $S^N = s^N_1 s^N_2 \ldots s^N_{l_N}$ using the merge operation and allowing to merge w characters of the S^1 string, w of the S^2,..., and w of the S^N is:

$$\begin{aligned} T_D(i_1,\ldots,i_N) = & min(T_D(i_1 - r_1,\ldots,i_j - r_j,\ldots,i_N - r_N) \\ & + \delta(S^1_{i_1-r_1+1,i_1},\ldots,S^j_{i_j-r_j+1,i_j},\ldots,S^N_{i_N-r_N+1,i_N})) \end{aligned} \qquad (5)$$

where $0 \leq r_j \leq w$ and $1 \leq j \leq N$, T_D being initialized as: $T_D(0,\ldots,i_j,\ldots,i_N) = 0$ for $0 \leq i_j \leq l_j$, and $T_D(0,\ldots,i_j,\ldots,i_N) = \infty$ for $l_j + 1 \leq i_j \leq 2l_j$. Then $D_c(S^1,\ldots,S^N)$ is defined as:

$$D_c(S^1,\ldots,S^N) = min(T_D(k_1,\ldots,k_j,\ldots,k_N)) \qquad (6)$$

where $l_j \leq k_j \leq 2l_j - 1$.

3 A Mean String Algorithm for Strings Representing Polygonal Shapes

In the present work, without loss of generality, shapes have been polygonally approximated. A shape is represented by an *attributed string* $S = s_1 ... s_n$, introduced by Tsai and Yu [11], where each line segment s_i is represented by

its length l_i and its angle ϕ_i, which is computed with respect to a reference horizontal line segment. The polygons are compared using the cyclic string edit distance, and the costs of the edit operations are defined as a weighted sum of an angle cost and a length cost, similarly to [12]. We have used the following cost function in such a way that a string a_1a_2 is merged into a symbol a by defining the l_a and ϕ_a attributes in the following way:

$$l_a = l_{a_1} + l_{a_2} \tag{7}$$

$$\phi_a = \phi_{\vec{a_1}+\vec{a_2}} \tag{8}$$

where $\vec{a_1}$ is the vector defined by the segment a_1 and $\vec{a_2}$ is the vector defined by the segment a_2. Once the symbol a has been defined, the merging cost is computed by the following equation:

$$c(a_1a_2 \to a) = \frac{|\phi_a - \phi_1|}{2\pi}\frac{l_1}{l_a} + \frac{|\phi_a - \phi_2|}{2\pi}\frac{l_2}{l_a} \tag{9}$$

The previous equations can be iteratively extended to merging substrings longer than two symbols. Note that the substitution, deletion and insertion are particular cases of the block substitution using the merging operation. Then, the total cost to substitute a whole sequence of symbols by another one $(A_{i,j} \to B_{k,l})$ is computed as follows:

$$c(A_{i,j} \to B_{k,l}) = c(A_{i,j}, a) + c(B_{k,l}, b) + c(a, b). \tag{10}$$

a and b being the symbols obtained after applying the merge operation $A_{i,j} \to a$ and $B_{k,l} \to b$.

The mean string algorithm has some particular issues when 2D shapes are polygonally approximated and represented by attributed strings. First of all, the alphabet for the symbols is not finite. As the symbols are edges, with the length l and the angle ϕ as attributes, there is an infinite number of possible values. It does therefore make no sense to look for a string in the set of possible words that minimize the cost, as this set is infinite. It makes more sense to *compute* a string which minimizes the cost. In our case, since shapes are polygonally approximated and are represented by attributed strings, the problem is transformed into computing a geometrical mean among polygons. Informally, the algorithm can be explained as follows:

Given two strings A and B, the string matching algorithm is computed, and hence the edit distance table T_D. Each cell $T_D(j, l)$ is associated with the selected edit operation $A_{i,j} \to B_{k,l}$ that involves the substitution of an open polygon $A_{i,j}$ of A by an open polygon $B_{k,l}$ of B. The idea is to compute, for each cell, the mean substring C between $A_{i,j}$ and $B_{k,l}$. In this context, the computation of the mean string of a set of strings: $S = \{S^1, S^2, \ldots, S^N\}$ is equivalent to computing the mean shape M whose edges have as attribute values the mean values computed among the edges of the set of strings S.

The algorithm follows the same basic structure as the one used to compute the edit distance table T_D, see equation 5. But now it needs an additional table T_M with the same dimensions as T_D to store the mean substring C explained above. Then, while it computes the cost in a cell $T_D(i_1, \ldots, i_N)$, it also computes the substring of the mean string and keeps it in $T_M(i_1, i_2, \ldots, i_N)$; this substring is the mean among all the substrings $S^j_{b_j,e_j}$ which the algorithm has decided to merge and to substitute at that point of the table T_D. At the end, for each cell $T_D(i_1, i_2, \ldots, i_N)$, a $T_M(i_1, i_2, \ldots, i_N)$ has been computed with the set of edges of the mean substring at that point. The mean string is obtained by the $T_M(i_1, i_2, \ldots, i_N)$ associated to the cells involved in the minimum cost edit sequence.

The key of this algorithm is the step to compute the substring of the mean string $M(i_1, i_2, \ldots, i_N)$ in each cell of the table T_M. To compute it, we need to know which substrings, of the N strings we are comparing, are selected to be merged and substituted among them. Then, if we are computing the mean substring in $T_M(i_1, i_2, \ldots, i_N)$, we have to know which edit operation has been selected in $T_D(i_1, i_2, \ldots, i_N)$; if it is the one coming from $T_D(r_1, r_2, \ldots, r_N)$, then the selected substrings from which we have to compute the mean substring are $S^1_{r_1,i_1}, S^2_{r_2,i_2}, \ldots, S^N_{r_N,i_N}$. As a substring is in fact an open string, and in order to simplify the notation, we will from now call them strings and will denote them as $S'^1, \ldots, S'^N$. Then, it is necessary to compute their mean. The idea is to see the N strings as piecewise linear functions. Given a string S, let us denote as F_S the piecewise linear function induced by the string S. To compute the mean among the set of strings, we compute the mean function among all the piecewise linear functions. The problem is to have a common parameterization for the linear functions, so that we can compute their mean. A first attempt could be to use as parameter the coordinate axis x, $F_{S'^i} : \mathbb{R} \to \mathbb{R}$, but when trying to make this transformation, we may end up in certain cases having two y-values for some x-values, which means that the problem cannot be formulated in terms of a mean function computation. We illustrate this problem by the example shown in Fig. 1. For that reason, we implemented a method that bases the common parameterization on the length percentage of each edge from the total set. This method to transform the set of edges into a piecewise linear function takes as parameter the length percentage of each segment relative to the total length of the set of edges $F_{S'^i} : [0, 1] \to \mathbb{R}^2$. These functions are defined for each of their pieces by the equation of the line representing the edge for that length percentage, and goes from the length percentage in [0,1] to the (x, y) of the edge that has the point corresponding to this percentage. As this percentage goes from 0 to 1 in all the edge sets for each percentage, we will have a correspondence, that is we have a common parameterization. Then, to compute the mean string among the N strings, we

just have to compute the mean function among the N functions:

$$F_M(lp) = \frac{\sum_{i=1}^{N} F_{S'^i}(lp)}{N} \tag{11}$$

M is the mean string of the N strings $S'^1, S'^2, \ldots, S'^N$, and it is formed by the line segments defined by the piecewise linear function $F_M(lp)$.

The steps of the algorithm explained above are listed below, with a graphical explanation presented in Fig. 2, which explains the case of computing the mean between two strings A and B.

(1) Transform all the S'^i into piecewise linear functions $F_{S'^i}(lp)$ defined by the line equation of each segment, where lp is the corresponding length percentage, see Fig. 2(a).
(2) Compute the mean between the N functions.

$$F_M(lp) = \frac{\sum_{i=1}^{N} F_{S'^i}(lp)}{N} \tag{12}$$

Then M will be the set of segments defined by $F_M(lp)$, and the mean set of segments of all the S'^i, see Fig. 2(b).

4 Complexity analysis

The presented problem of computing the mean shape among a set of shapes has a high computational cost, as it is necessary to compute two tables T_D and T_M, and from equation 5 it is possible to deduce that the computation of T_D is an exponential problem with the number of strings. In general the computation of the mean string is a NP-hard problem, as proved in [5]. Due to this high complexity, we have made some changes in the algorithm to make it useful. For that purpose, instead of computing the table T_D and T_M for N shapes, we compute the mean shape among them in a progressive way, that is computing it two by two and adding one more shape at each step. This modification leads to a sub-optimal solution, as the matching among the edges forming the shapes is not always exactly the same when we compute it among all the shapes at the same time than when we compute it shape by shape. The problem in that adaptation is to guarantee that each shape will have the same weight in the process. For that purpose, we need to keep a record, for each mean shape, of the number of shapes it represents and of the partial addition of the piecewise linear functions that are describing it. In that way, we can ensure that when two mean shapes are grouped, the one representing more shapes will have more weight when computing the new mean between them.

Then, each edge has as attributes not only the length l and the angle ϕ, but also the partial addition of all the edges it represents, treated as piecewise linear functions, and the number of these edges. With all these attributes, we can maintain the partial addition at each step, and we can compute the mean string of a set at each step of the algorithm. This behaviour also allows us to parallelize the computation of the mean string from a big set of shapes. An example of this incremental way to compute the mean shape is presented in Fig. 3, where we can see the shapes S^1, S^2, S^3, S^4 and S^5 and how we can compute their mean. First, the mean $S^{1,2}$ and the mean $S^{3,4}$ are computed. Then, the mean shape $S^{1,...,4}$ is computed. At the end, only one shape S^5 is added to the mean and we have the mean $S^{1,...,5}$. We can thus see that S^5 has less weight than $S^{1,...,4}$, as it is representing only one shape, while the other already represent four shapes. The crossed-over shape would be the mean shape between $S^{1,...,4}$ and S^5 if both would be considered as representing only one shape.

Till this point, we have explained how to reduce the problem by means of an approximation: instead of computing a big T_D, compute several smaller $T'_D s$, one for each comparison between two shapes. But it is also possible to reduce the number of operations we make to compute T_M. In fact, it is not necessary to compute all the table T_M. It is enough to first compute the edit distance table T_D, and then the mean string associated with the underlying edit sequence. Thus, only the substrings we need for the mean string are computed, and we do not make lots of useless computations. For the computational complexity analysis of the algorithm we should first make the following considerations, considering that N is the number of polygons, and n their number of edges. First, the minimum cost string edit sequence between two string-encoded shapes is computed in $O(N^2 logn)$ [19]. Second, the mean string between two strings, following the method described in Section 3, can be computed in $O(n)$. Thus, the incremental computing of the mean among N strings requires a worst case bound of $O(Nn^2 logn)$. Our suboptimal procedure to compute the mean string is even lower than the complexity of the set median string computation that is bounded by $O(N^2 n^2 logn)$.

5 Results

In this section, two kinds of experiments are shown. In the first, the algorithm has been applied to polygonal approximations of real shapes taken from the Rutgers tools database [1] . In the second, a texture segmentation and representation in architectural drawings is done by means of the mean shape.

[1] This database can be downloaded from:
http://www.cs.rutgers.edu/pub/sven/rutgers-tools/

In the first experiment the results of computing the mean shape among ten images taken two by two are presented in Table 1. At each position in the table, we show the mean between the corresponding row and column polygon. Two points should be noticed to visually assess the performance of the algorithm. First, the result of computing the mean between a polygon and itself, see the diagonal, is the same polygon. Second, look at three particular details on the polygons: the size, the curvature of their handles and the opening of their jaws. Let us further examine the mean between $Pl6$ and $Pl5$, two polygons quite different according to the above features. The obtained mean presents an opening in its jaws smaller than in $Pl6$ and bigger than in $Pl5$, the same happens with the roundness of its jaws, they are more round than the ones in $Pl6$ and less than the ones in $Pl5$. Although this behaviour is found in all the cases sometimes we can see that the results are not symmetric, for example the mean shape between $Pl7$ and $so42$ is not exactly the same than the one in the other sense between $so42$ and $Pl7$, this is because the string edit distance is not symmetric and gives us different edge alignments. The computation of the mean string is a part of a more general work focused on graphical document analysis [16,17]. The second kind of experiment has been done in this domain. The aim of this work is to segment the textured areas in a graphical document and to represent them in a compact way. We are considering structured textures consisting of polygonal shapes that are placed following a rule. To segment the textured areas, we cluster similar polygons appearing in the document, computing their similarity by using the string edit distance explained in this paper. At any intermediate step of the clustering algorithm, each cluster consists of a set of polygons and is represented by the mean among them. The decision of whether two clusters have to be merged is made according to the string edit distance between their representative polygons. At the end, we have the different textured areas segmented and for each one a part of its representation is the mean polygon representing all the polygons forming the texture. We have tested our algorithm clustering similar shapes in line drawings. In this clustering process the well behaviour of the computation of the mean shape is the key to obtain good results. Figure 4 shows an example where a drawing is printed in Fig. 4(a) and bold lines delineate the three different kinds of segmented textures. The first segmented texture is formed by vertical tiles represented by the computed mean shape in Fig. 4(b), it appears in the bottom left area of the image. The second segmented texture is formed by horizontal tiles and represented by the computed mean shape in Fig. 4(c), it is fragmented in three different areas in the top of the image the bottom right and a little triangle on the left. The last textured area is in fact a textured symbol, the stairs that are characterized by a set of steps which are rectangles placed in columns, the computed mean shape representing these steps is in Fig. 4(d). In all the cases, we can see that the clustering algorithm is able to join the similar shapes using the mean shape algorithm, and that at the end we have one representative shape for each different texture.

We should notice that the computed mean shapes can have a large number of edges. This is because a fragmentation of the edges of the computed mean shape occurs. When we compute the mean shape progressively—that is adding more and more shapes to be represented—we are actually adding at each step one more shape, but we are also fragmenting the edges of the mean shape. This can lead to an increment in the string edit distances from the mean shape to all the others. This is because, when we are comparing shapes having a different partition degree, the one with more edges has to be merged, and the merge operation has a cost. It is actually a "resolution" problem, as the mean shape is represented with a higher "resolution" or detail level than the original shapes, so that their distances are not comparable. To solve this problem, we must modify the cost of the merge operation, whenever the mean shape fragmentation requires it, or alternatively, we should transform the mean shape until it has the same "resolution" as the original ones.

6 Conclusions

In this paper, we have presented a sub-optimal algorithm to compute the mean of a given set of shapes. Since shapes are represented by cyclic strings attributed by geometric information, the mean shape computation is actually formulated in terms of a mean cyclic string problem. The key idea of the algorithm has been to transform the strings representing shapes in piecewise linear functions, using as common parameterization the length percentage, and then computing the mean of those piecewise linear functions. We have incorporated this idea into the well-known dynamic-programming algorithm to compute the minimum cost edit sequence between strings.

The contributions of our work should be stated from the practical point of view. First, a number of theoretical mean string algorithms can be found in the literature, but they often assume a finite alphabet of symbols. However, our algorithm reformulates the classical ones for strings representing shapes, i.e. for strings conveying geometrical attributes associated to symbols belonging to a non finite alphabet. Secondly, the computational cost of the mean string algorithm has been proven to grow exponentially in the number of strings. We have reduced this cost near a polynomial bound by defining an approximate solution that computes the mean string in an incremental way. On the other hand, the optimality of the solution also depends on the definition of the string edit operations in terms of the geometrical attributes, especially the merge operation.

The above considerations have been experimentally evaluated. As the distance used in equation 2 is not a real distance, but an approximation, a full quantitative evaluation of this algorithm is not feasible. However the segmentation

process based on this mean string computation has been proven to be very useful in the context of complete document analysis systems, such as the architectural drawing interpretation system described in [18]. In this framework, once a texture consisting in a regular repetition of similar shapes has been detected, the mean shape has been computed as the representative to compactly represent the textured area. In this context, the test has been applied on more than thirty documents with different types of textures and the results have been considered good enough according to the a priori expected segmentation and representation of textures. To asses the performance of the algorithm we have used a ground truth consisting of sets of this real shapes from architectural documents and also sets of synthetic shapes. The mean shape has been computed between two shapes or among a group of shapes and, in all the experiments, the resulting shape agrees the visually expected results allowing a correct segmentation and representation.

References

[1] H. Bunke and U. Bühler, Applications of approximate string matching to 2d shape recognition, *Pattern Recognition* **26** (1993) 1797–1812.

[2] H. Bunke and A. Kandel, Mean and maximum common subgraph of two graphs, *Pattern Recognition Letters* **21** (2000) 162–168.

[3] H. Bunke, A. Münger, and X. Jiang, Combinatorial search versus genetic algorithms: A case study based on the generalized median graph problem, *Pattern Recognition Letters* **20** (1999) 1271–1277.

[4] C.D.Martínez-Hinarejos, A. Juan, and F. Casacuberta, Use of Median String for Classification, *15th International Conference on Pattern Recognition ICPR2000* (Barcelona, Spain) **2** (2000) 907–910.

[5] C. de la Higuera, F. Casacuberta, Topology of strings: Median string is NP-complete, *Theoretical Computer Science* **230** (2000) 39–48.

[6] T.Kohonnen, Median strings, *Pattern Recognition Letters* **3** (1985) 309–313.

[7] T. Cootes, C. Taylor, and J. Graham, Active shape models- their training and application, *Computer Vision and Image Understanding* **61** (January 1995) 38–59.

[8] X. Jiang, A. Münger, and H. Bunke, Syhthesis of representative graphical symbols by computing generalized median graph, *3rd. International asociation for pattern recognition workshop on Graphics Recognition GREC'99* (Jaipur, India, 1999) 187–194.

[9] J. B. Kruskal, An overview of sequence comparison: Time warps, string edits, and macromolecules, *SIAM Review* **25(2)** (1983) 201–237.

[10] D. Lopresti and J. Zhou, Using consensus sequence voting to correct ocr errors, in: A. Lawrence Spitz and A. Dengel, eds., *DAS'94—IAPR workshop on Document Analysis Systems. Series in machine perception artificial intelligence, World Scientific* (1994) 157–168.

[11] W. Tsai and S. Yu, Attributed string matching with merging for shape recognition, *Proceedings of 7th. International Conference on Pattern Recognition*, (Montreal, Canada, 1984) 1162–1164.

[12] Y. Tsay and W. Tsai, Model-guided attributed string matching by split-and-merge for shape recognition, *International Journal of Pattern Recognition and Artificial Intelligence* **3(2)** (1989) 159–179.

[13] N. Ueda and S. Suzuki, Learning visual models from shape contours using multiscale convex/concave structure matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15(4)**(April 1993) 337–352.

[14] E. Valveny, Reconocimiento de símbolos gráficos a mano alzada mediante modelos deformables en un entorno Bayesiano, *PhD thesis, Universitat Autònoma de Barcelona* (September 1999).

[15] R. Wagner and M. Fischer, The string-to-string correction problem, *Journal of the Association for Computing Machinery* **21(1)**(1974) 168–173.

[16] J. Lladós and G. Sánchez and E. Martí, A string-based method to recognize symbols and structural textures in architectural plans, in: K. Tombre and A.K.Chhabra, eds., *Graphics Recognition, Algorithms and Systems. Lecture Notes in Computer Science, Springer-Verlag*, **1389**, (1998) 91–103.

[17] Ph. Dosch and C. Ah-Soon and G. Masini and G. Sánchez and K. Tombre, Design of an Integrated Environment for the Automated Analysis of Architectural Drawings, in: S.-W. Lee and Y. Nakano, eds., *Document Analysis Systems: Theory and Practice. Selected papers from DAS'98. Lecture Notes in Computer Science, Springer-Verlag*, **1655**, (1999) 295–309.

[18] Ph. Dosch, G. Masini and K. Tombre, Improving Arc Detection in Graphics Recognition, in Proceedings of 15th International Conference on Pattern Recognition, Barcelona (Spain), **2**, September (2000) 243–246.

[19] M. Maes, On a Cyclic String-to-String Correction Problem, *Information Processing Letters*, **35**,(1990) 73–78.

[20] A. Marzal and S. Barracina. Speeding up the computation of the edit distance for cyclic strings. *15th Conference on Pattern Recognition (ICPR2000)*, **2**, (2000) 895–898.

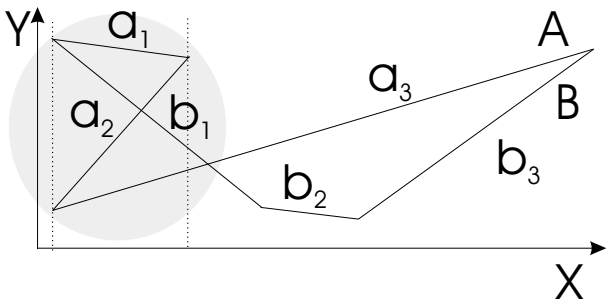
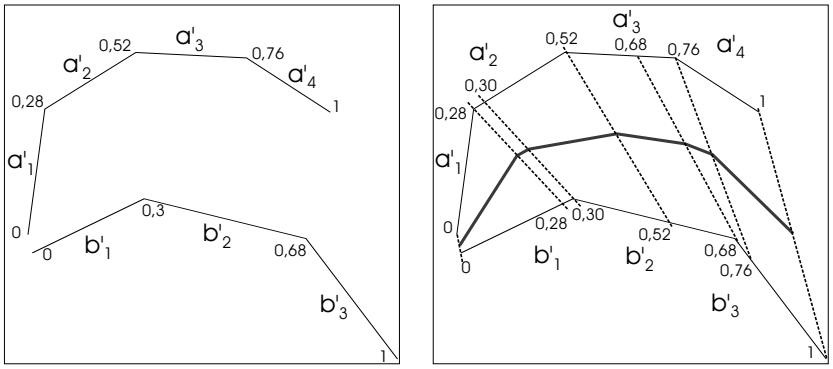


Fig. 1. The problem using the parameterization in terms of the x value.



(a) (b)

Fig. 2. Phases of the computation of the mean string M: (a)Transformation of A and B into two piece wise linear functions using the length percentage as a common parameterization. (b)Mean string, in bold lines.

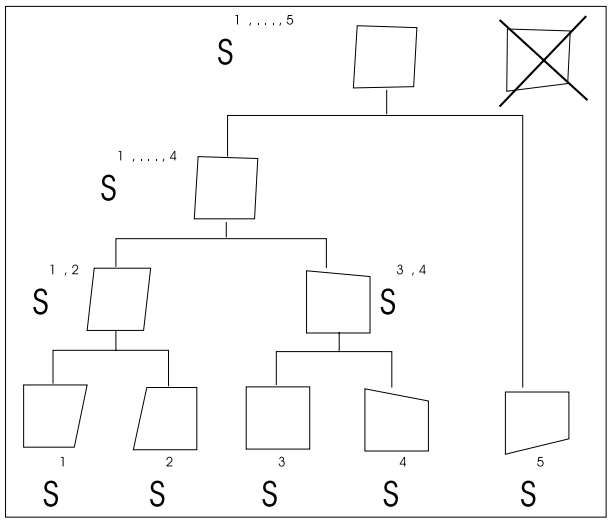


Fig. 3. The progressive way to find the mean shape.

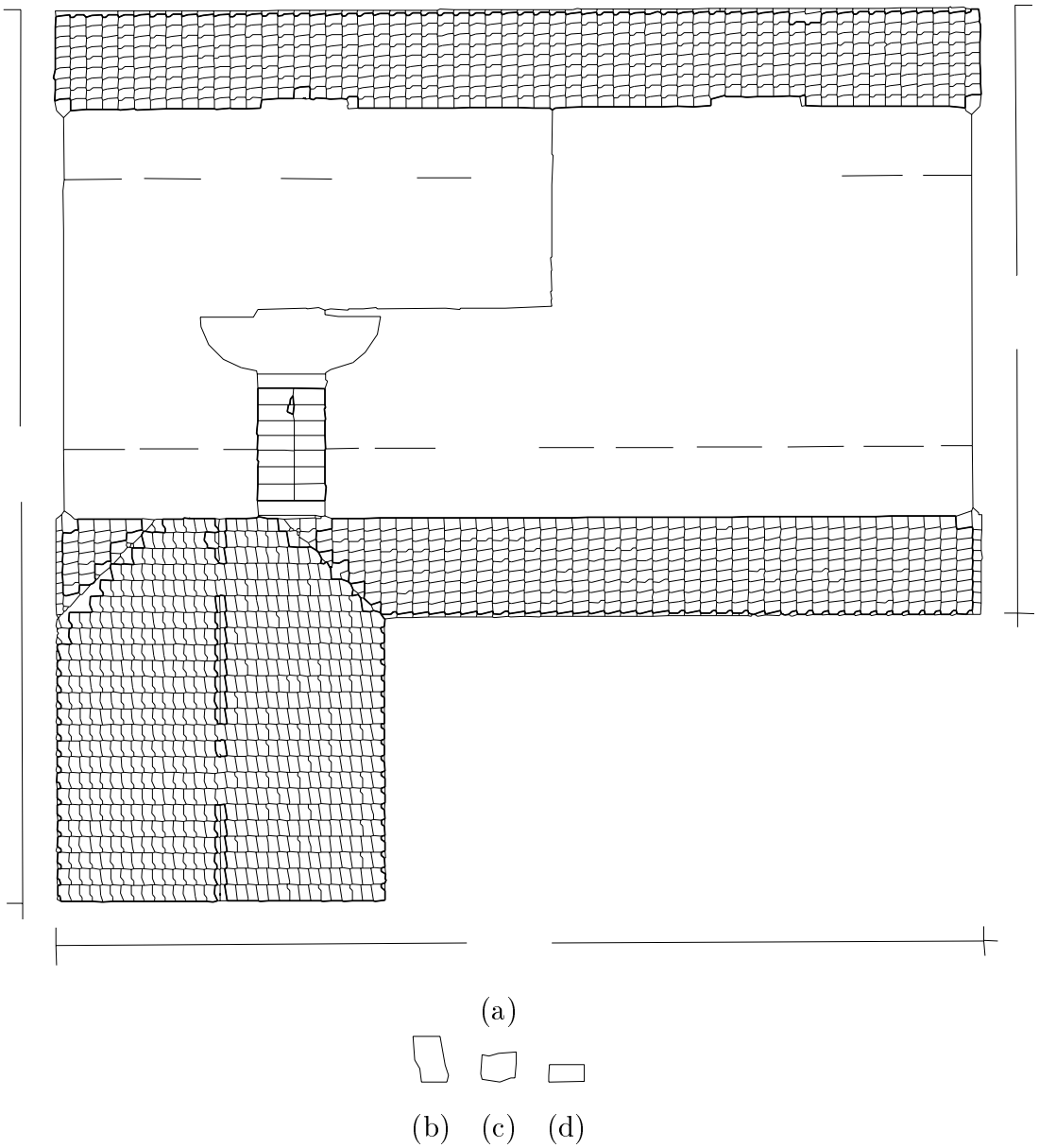


Fig. 4. (a) Drawing and—in bold lines—borders of three different clustered textured areas. (b) Detail of the mean shape for vertical tiles. (c) Detail of the mean shape for horizontal tiles. (d) Detail of mean shape for steps in the stairs.

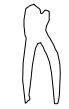
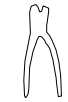
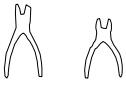
	Pl6	so41	so42	Pl5	ori1	ori3	pl1	Pl3	Pl7	Pl8
Pl6										
so41										
so42										
Pl5										
ori1										
ori3										
pl1										
Pl3										
Pl7										
Pl8										

Table 1
Mean string of 10 real shapes from Rutgers tools database.