CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Repositori Institucional de la Universitat Jaume I

Título artículo / Títol article:	On hidden Markov models and cyclic strings for shape recognition
Autores / Autors	Vicente Palazón-González Andrés Marzal Juan M. Vilar
Revista:	Pattern Recognition Volume 47, Issue 7, July 2014, Pages 2490–2504
Versión / Versió:	Pre-print
Cita bibliográfica / Cita bibliogràfica (ISO 690):	PALAZÓN-GONZÁLEZ, Vicente; MARZAL, Andrés; VILAR, Juan M. On hidden Markov models and cyclic strings for shape recognition. *Pattern Recognition*, 2014, vol. 47, no 7, p. 2490-2504.
url Repositori UJI:	http://hdl.handle.net/10234/124503

On Hidden Markov Models and Cyclic Strings for Shape Recognition

Vicente Palazón-González, Andrés Marzal, Juan M. Vilar

Universitat Jaume I, Dept. Llenguatges i Sistemes Informàtics and Institute of New Imaging Technologies, Castellón de la Plana, Spain

Abstract

Shape descriptions and the corresponding matching techniques must be robust to noise and invariant to transformations for their use in recognition tasks. Most transformations are relatively easy to handle when contours are represented by strings. However, starting point invariance is difficult to achieve. One interesting possibility is the use of cyclic strings, which are strings that have no starting and final points. We propose new methodologies to use Hidden Markov Models to classify contours represented by cyclic strings. Experimental results show that our proposals outperform other methods in the literature.

Keywords:

Hidden markov models, cyclic strings, shape recognition.

1. Introduction

Shape recognition is a very important problem with applications in several areas including industry, medicine, biometrics and even entertainment.

In a shape classifier, shapes can be represented by their contours or by their regions [1]. However, contour based descriptors are widely used as they preserve local information, which is important in the classification of complex shapes.

Dynamic Time Warping (DTW) [2] is being increasingly applied for shape matching [3–6]. A DTW-based dissimilarity measure is a natural option for optimally aligning contours, since it is able to align parts as well as points and it is robust to deformations. Hidden Markov Models (HMMs) [7] are also used for shape modelling and classification [8–14]. They are stochastic generalizations

Email addresses: palazon@lsi.uji.es (Vicente Palazón-González), amarzal@lsi.uji.es (Andrés Marzal), jvilar@lsi.uji.es (Juan M. Vilar)

of finite-state automata, where transitions between states and generation of output symbols are modelled by probability distributions. HMMs have some of the properties of DTW matching and also provide a probabilistic framework for training and classification with the advantages that it entails. A statistical model is much more compact in terms of storage and its temporal cost in classification is not related with the quantity of training samples unlike in distance-based methods.

Shape descriptors, combined with shape matching techniques, must be invariant to many distortions, including scale, rotation, noise, etc. Most of these distortions are relatively easy to deal with. However, no matter the representation, invariance to the starting point is difficult to achieve.

We can find the following solutions in the literature for obtaining invariance to the starting point in the context of HMMs: the election of a reference rotation [8], a circular topology [10] and using an ergodic model [12–14], so that training solves the problem.

A contour can be transformed into a string by choosing an appropriate starting point with a reference rotation [8], but this election is a heuristic which is unreliable in unrestricted scopes. In [12–14], the authors use ergodic topologies, which have the consequence that different non-consecutive parts of the contour can be explained by the same state. This makes recognition a more complex problem. Left-to-right topologies seem better suited for recognizing strings. In [10], the authors propose a circular topology to model cyclic strings. Their structure removes the need for a starting point: a cyclic string can be segmented to associate consecutive states to consecutive segments in the strings, but there is no assumption about which state is the first or the last one. As we will see this and the ergodic topology have similar problems.

The best solution to this problem is to consider every possible starting symbol of the string. The concept of cyclic string arises here. A cyclic string is a string of symbols or values that has neither beginning nor end, i.e., a cyclic string models the set of every possible cyclic shift of a string.

So the question is: how can we train HMMs for cyclic strings? There is a time order, but we do not know where the strings begin. HMMs only can generate ordinary strings and not cyclic strings. To overcome this problem, in this paper, we will propose new methodologies to properly work with HMMs in order to classify cyclic strings. Preliminary work on this problem appears in [15].

This document is organized as follows. In Section 2, HMMs are revisited. In Section 3, the

drawbacks of other methods in the literature are pointed out and the main problem to solve is defined. The training of cyclic strings is presented in Section 4. Methods for speeding up cyclic training and recognition with Linear HMMs are presented in Section 5. A better heuristic for choosing a starting point is presented in Section 6. In Section 7, some considerations about the computational complexity of the proposed methods are made. In Section 8, experimental results on image classification tasks for several databases compare the different methods. Finally some conclusions are commented in Section 9.

2. Hidden Markov Models

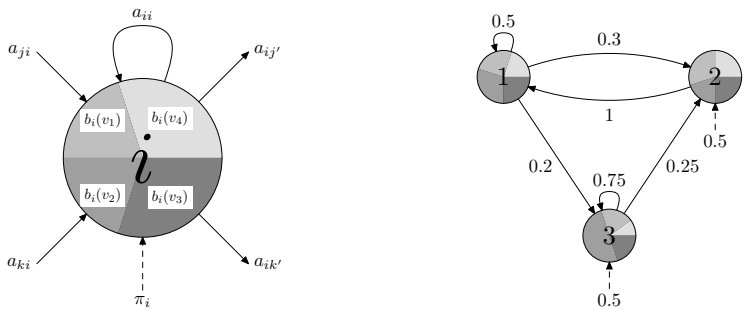
An HMM [7] contains a set of states, $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$, where each one has an associated probability distribution for the emission of symbols. In each instant, t, a state produces an observable event that only depends on that state. Similarly the transition from one state to another is a random event that only depends on the state from which the transition starts.

Definition 1. *Let $\Sigma = \{v_1, v_2, \ldots, v_w\}$, be an alphabet (the set of observable events is discrete and finite), a* discrete HMM *with n states is a triplet (A, B, π), where:*

- *$A = \{a_{ij}\}$, for $1 \leq i, j \leq n$, is the matrix of transition probabilities (a_{ij} is the probability of being in state j at instant $t+1$ conditioned on being in state i, at instant t);*
- *$B = \{b_{ij}\}$, for $1 \leq i, j \leq n$ and $1 \leq j \leq w$, is the matrix of observation probabilities (b_{ij}, or $b_i(v_j)$, is the probability of observing v_j, being in state i);*
- *$\pi = \{\pi_i\}$, for $1 \leq i \leq n$, is a probability distribution for initial states (π_i is the probability of being in state i when $t = 1$).*

Besides, the following conditions must be satisfied for all i: $\sum_{1 \leq j \leq n} a_{ij} = 1$, $\sum_{1 \leq j \leq w} b_{ij} = 1$ and $\sum_{1 \leq i \leq n} \pi_i = 1$.

Figure 1a shows a graphical representation of a discrete state and Figure 1b shows a complete discrete HMM.



(a) (b)

Figure 1: (a) An HMM state that can emit any of four symbols according to the probability distribution represented by a pie chart. (b) A complete HMM. The dotted lines represent the probability distribution of the initial states.

If the observable elements of the strings are continuous, their probability is usually assumed to follow a normal distribution:

$$b_i(v) = \mathcal{N}(v; \mu_i, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{v-\mu_i}{\sigma_i})^2}.$$

If the elements of the strings have several dimensions, an n-dimensional normal distribution can be used:

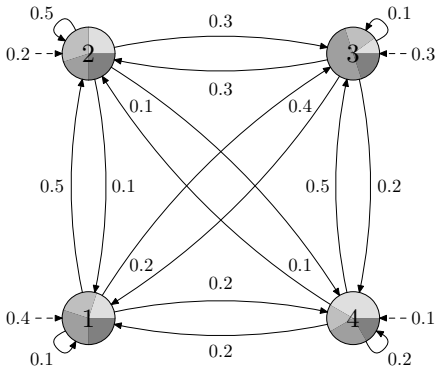
$$b_i(v) = \mathcal{N}(v; \mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} e^{-\frac{1}{2}(v-\mu_i)'\Sigma_i^{-1}(v-\mu_i)}.$$

Also mixtures of normals are used for multimodal distributions [7].

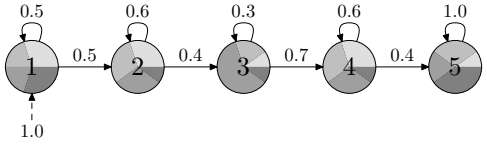
2.1. Topologies

The number of states, n, and the state transition probabilities which are not zero, $a_{ij} \neq 0$, define the topology of the HMMs. Topologies impose important restrictions over the stochastic process and produce different behaviours in the models. There are a large number of topologies depending on the application domain. Two of the most commonly used are: ergodic models and left-to-right models (or temporal).

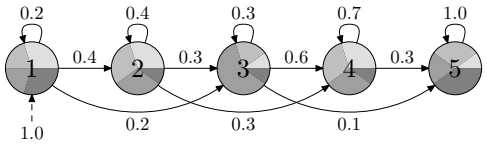
The matrix of transition probabilities of the ergodic models has no null entries, therefore, it is possible to reach every state from any other (see Figure 2a). For some applications, where recognition requires a certain temporality (as in speech recognition), left-to-right topologies obtain better results. In these topologies, $a_{ij} = 0$ for $j < i$. Figures 2b and 2c show two particular cases of these topologies. More concretely, in Figure 2b there is a linear left-to-right HMM, where from



(a)



(b)



(c)

Figure 2: Examples of topologies. (a) Ergodic topology with four states. (b) Linear left-to-right topology. (c) Bakis left-to-right topology.

a state it is possible to transit to itself or to the next state. And in Figure 2c there is a Bakis left-to-right topology.

In the following, instead of a probability distribution for the initial states, we will have a single non-emitting initial state, S_0, which is not reachable from any other state. HMMs are then simply defined by $\lambda = (A, B)$. Thus, with this new way of defining the Markov models, for example, the model in Figure 1b would have a non-emitting state S_0 with two transitions, one to state S_2 with a probability 0.5 and the other one to state S_3 with the same probability.

2.2. Training of HMMs

Given a set of observations, the training problem consists in finding a HMM that models them adequately. Usually, the problem is approached using a maximum likelihood approach. That is, given a string $x = x_1 x_2 \dots x_m$, the aim is to find a model $\lambda(A, B)$ that maximizes $P(x|\lambda)$.

If a topology is assumed, the well known *forward-backward* algorithm [7] can be used to efficiently estimate the parameters of the model. Unfortunately, there are not known effective methods to also infer the topology.

The basic idea of the *forward-backward* algorithm is to compute two probabilities:

- The *forward* probability: the probability of observing a prefix of the input string and ending in a given state; and
- the *backward* probability: the probability that a given tail of the string is produced by starting in a given state.

Conventionally, the forward probability is represented by $\alpha_t(i)$ with the meaning

$$\alpha_t(i) = P(x_1 \dots x_t, Q_t = S_i|\lambda).$$

Analogously, the backward probability is represented by $\beta_t(i)$ with the meaning

$$\beta_t(i) = P(x_{t+1} \dots x_m|Q_t = S_i, \lambda)$$

There are well known recursive procedures to compute both of them [7]. Once the values of α and β are known they can be used to reestimate the transition and emission probabilities. This is done using Expectation Maximization [16, 17]. The new value of a_{ij} is the quotient between the

expected number of times a transition happened from state S_i to S_j divided by the expected number of times the state S_j was visited. Analogously, the new value of $b_i(v_j)$ is the quotient between the expected number of times the state S_i emitted the symbol v_j and the expected number of times the state S_j was visited. As mentioned, these expected values can be easily computed from α and β, and the details can be seen in [7]. We will see next how this method can be modified to cope with cyclic strings, but before, let us briefly mention the computation of the probability of a string given the model, the decoding problem and the Viterbi approach to training.

When a string x is given, it is interesting to know the probability that this string was generated by the model. Given that x could have been generated by any sequence of states, the sought probability is

$$P(x|\lambda) = \sum_{Q \in \mathcal{S}^m} P(x|Q,\lambda)P(Q|\lambda). \tag{1}$$

It is easy to prove that for any t

$$P(x|\lambda) = \sum_{i=0}^{n} \alpha_t(i)\beta_t(i).$$

In particular, setting $t = m$,

$$P(x|\lambda) = \sum_{i=0}^{n} \alpha_m(i).$$

Usually, decoding is understood as the problem of finding the most probable sequence of states given the output of the model. This sequence of states can be interpreted as an alignment between the output string, x, and the states of the model so that it explains which state produced which symbol with maximum probability. This can be seen as changing the summation in 2 for a maximization, i.e.:

$$\hat{P}(x|\lambda) = \max_{Q \in \mathcal{S}^m} P(x|Q,\lambda)P(Q|\lambda). \tag{2}$$

The Viterbi algorithm [7, 18] can be used to efficiently find this value using an approach analogous to the computation of α and β.

When it is expected that the most probable path contains a significant fraction of the total probability mass of the string, it is possible to reestimate the model using only those paths for estimating the counts of the transition. This is usually known as Viterbi training.

3. Defining the Problem of Cyclic Strings with HMMs

As we commented before, the following methods are used for obtaining the starting point invariance: use of an ergodic model [12–14] where training solves the problem, a circular topology [10] and the election of a reference rotation [8].

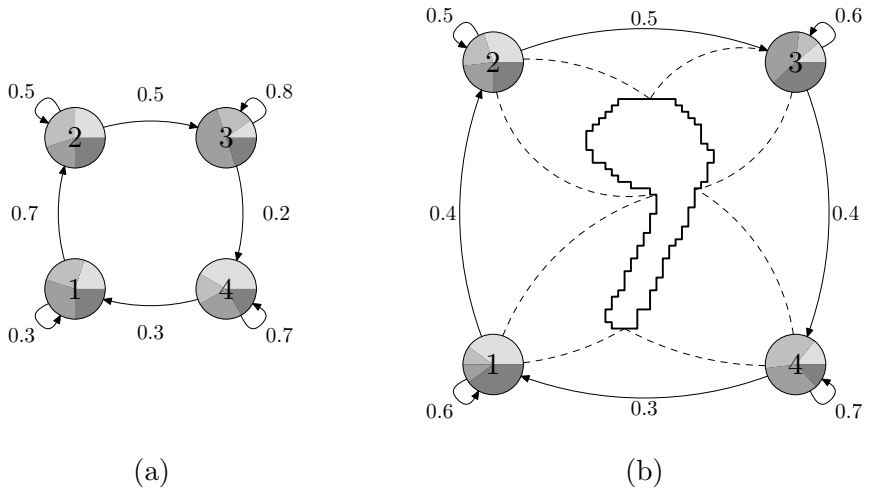
3.1. Ergodic Models

Observe that for modelling a class of shapes we must use an HMM with enough states for considering all possible variations of the class. Many works use ergodic topologies, which have some problems. In these topologies, it is possible to visit a state more than once without using self transitions (see Figure 2a). Ergodic models do not impose restrictions in the order of the strings of observations. When the string of observations is temporal or an order exists (as in shape contours), these topologies do not fully use the sequential or temporal information of the data and many states are used to explain multiple observations from different parts along the contour. This makes recognition a complex problem. Moreover, the training process in these models is very sensitive to initialization and to local estimation of parameters.

3.2. Circular Topologies

From the previous observations, left-to-right topologies seem more suitable. These topologies do not allow to visit states that are to the left of the current one, forcing transitions to follow: $a_{ij} = 0$ for $j < i$. In left-to-right models there is an initial state and a final state. This way, the traversed sequence of states is forced to begin in the initial state (the leftmost one, Figure 2b and Figure 2c) and to transit to posterior states (that is to say, to the right) or to the current state. Thus, the sequence of states represents the passage of time. When a string of symbols is segmented, all the symbols of a segment are emitted by the same state, and consecutive segments are associated to consecutive states. Although these topologies usually have more states (they can be determined by the characteristics that protrude in the contour), their number of transitions is low, and consequently the overall complexity of algorithms is reduced.

As cyclic strings have neither beginning nor end, HMMs may seem inappropriate to model them. In [10], a circular topology is proposed to model cyclic strings, Figure 3a shows this topology, which can be seen as a modification of the left-to-right topology, where the last emitting state is



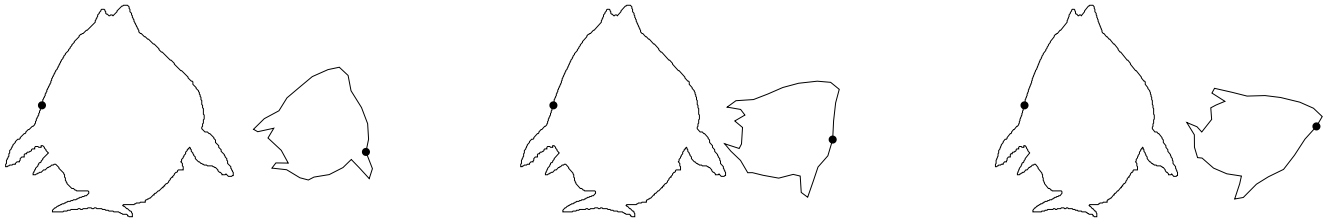
(a) (b)

Figure 3: (a) Circular topology as proposed in [10]. (b) The contour of a shape is segmented and each segment is associated to a state of the HMM. Ideally, each state is responsible for a single segment.

connected to the first one. This topology eliminates the necessity of defining a starting point: the cyclic string can be segmented for associating consecutive states to consecutive segments in the cyclic strings, but there is no assumption about which is the first or last segment (see Figure 3b); therefore, there is an analogy with left-to-right topologies. However, there is a problem that breaks this analogy: like in the case of ergodic models all the states can be reached from any state and we can finish in any of them. Therefore, the optimal path can contain non-consecutive repeated states and one state can be responsible of the emission of several non-consecutive segments of the cyclic string. Besides, it is possible to have an optimal path that does not visit all the states at least once.

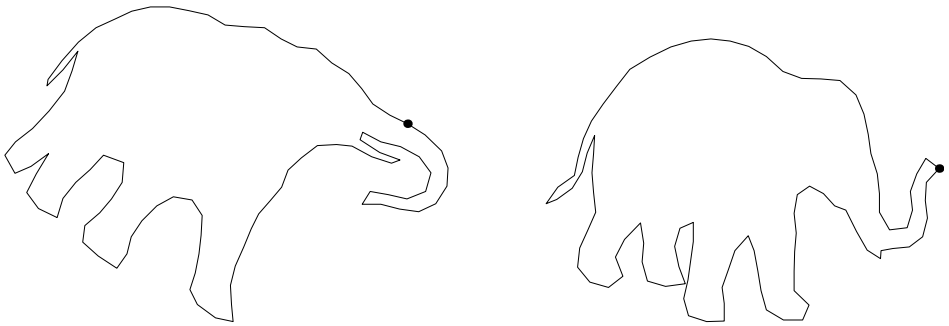
3.3. Reference Rotation

The basic idea of the election of a reference rotation [8, 19–21] is that, after normalization, all shapes have a canonical version with a "standard" rotation and starting point, and thus, they can be compared as if their representations were linear. But invariance is only achieved for different rotations and starting points of the same shape. Different shapes (even similar ones) may differ substantially in their canonical orientation and starting point. Figure 4 shows three perceptually similar figures (in fact, the second and third ones have been obtained from the first one by slightly compressing the horizontal axis) whose canonical versions are significantly different in terms of orientation and starting point. This problem frequently appears in shapes whose basic ellipse is



(a) (b) (c)

Figure 4: (a) Original shape and its canonical version using FDs [20, 21]. (b) The same shape compressed in the horizontal axis and its canonical version, which has a different rotation and starting point. (c) A slightly more compressed shape and its canonical version, which is also different.



(a) (b)

Figure 5: (a) Canonical version of an elephant with its trunk down. (b) Canonical version of an elephant with its trunk raised. Both canonical versions have been obtained by the method of least second moment of inertia [8, 19].

almost a circle. Besides, shapes of the same category with little differences can substantially alter the selection of the starting point. Figure 5 shows two elephants, one with its trunk down and the other with its trunk raised, this fact and other little differences modify the canonical rotation of the method of least second moment of inertia, and with it, the selection of the starting point.

3.4. Cyclic Strings

The most suitable solution for obtaining the invariance to the starting point is to use every possible starting point of the strings, that is to say, using cyclic strings. They can be defined as follows:

Definition 2 ([22]). *Let $x = x_1 \dots x_m$ be a string from an alphabet Σ. The* cyclic shift *$\rho(x)$ of a string x is defined as $\rho(x_1 \dots x_m) = x_2 \dots x_m x_1$. Let ρ^k denote the composition of k cyclic shifts and let ρ^0 denote the identity. Two strings x and x' are* cyclically *equivalent if $x = \rho^k(x')$, for*

some k. The equivalence class of x is $[x] = \{\rho^k(x) : 0 \leq k < m\}$ and it is called a cyclic string. *Any of its members is a representative of the cyclic string. For instance, the set $\{wzz, zzw, zwz\}$ is a cyclic string and wzz (or any other string in the set) can be taken as its representative.*

To achieve starting point invariance using cyclic strings we model the generation process as follows.

An HMM generated a string that later suffered a cyclic shift, but we do not know which one. That is to say, a model, λ, has generated a string, $x = x_1 x_2 \dots x_m$, that has suffered the transformation, $\rho^{k'}(x)$, for an unknown k'. We treat x as a cyclic string, $[x] = \{\rho^k(x) : 0 \leq k < m\}$, and we assume that all the cyclic shifts are equiprobable.

Thus, the evaluation problem can be solved with

$$\begin{aligned} P([x]|\lambda) &= \sum_{k=0}^{m-1} P(x|\lambda, k) P(k|\lambda) \\ &= \frac{1}{m} \sum_{k=0}^{m-1} P(x|\lambda, k) \\ &= \frac{1}{m} \sum_{k=0}^{m-1} P(\rho^k(x)|\lambda), \end{aligned} \tag{3}$$

that is, we must compute the probability for every possible cyclic shift and add them.

Similarly, the decoding problem can be solved with

$$\begin{aligned} \hat{P}([x]|\lambda) &= \max_{0 \leq k \leq m-1} \hat{P}(x|\lambda, k) P(k|\lambda) \\ &= \frac{1}{m} \max_{0 \leq k \leq m-1} \hat{P}(\rho^k(x)|\lambda) \propto \max_{0 \leq k \leq m-1} \hat{P}(\rho^k(x)|\lambda). \end{aligned} \tag{4}$$

The optimal sequence of states, $\hat{Q}$, will correspond to the cyclic shift that has the highest Viterbi score.

Initially, we adopt this score, $\hat{P}$, as an estimation of the real probability (3), because it is a very good approximation. Moreover, as we will see in Section 5, it is possible to considerably reduce the computational cost of this procedure, and then, to speed up recognition and training with linear topologies.

4. Cyclic Training

To solve the training problem (see Section 2) with cyclic strings we have to estimate the Markov model parameters, maximizing the probability of the observed cyclic strings. That is to say, our

objective is to maximize:

$$P(X|\lambda) = \prod_{l=1}^{L} P([x]^{(l)}|\lambda) = \prod_{l=1}^{L} \frac{1}{m^{(l)}} \sum_{k=0}^{m^{(l)}-1} P(\rho^k(x^{(l)})|\lambda), \tag{5}$$

where X is a set of cyclic strings, $X = \{[x]^{(1)}, [x]^{(2)}, \ldots, [x]^{(L)}\}$.

We will use an iterative procedure for obtaining the parameters of λ which maximize this function. First, we will set some arbitrary values for λ. Then, we will obtain new values of these parameters, for each iteration, using increasing transformations, applying the Baum-Eagon inequality [23]. It is guaranteed that the new estimated values increase the value of the objective function and, therefore, its convergence.

As we know that $\sum_{j=0}^{n} a_{ij} = 1$, for $0 \leq i \leq n$ and that (5) is a polynomial with respect to A. The new estimation, $\bar{a}_{ij}$, can be obtained with the Baum-Eagon inequality [23, 24] (applying logarithms to (5)):

$$\begin{aligned} \bar{a}_{ij} &= \frac{\frac{\partial \log(P(X|\lambda))}{\partial a_{ij}} a_{ij}}{\sum_{j=0}^{n} \frac{\partial \log(P(X|\lambda))}{\partial a_{ij}} a_{ij}} \\ &= \frac{\sum_{l=1}^{L} \frac{\partial P([x]^{(l)}|\lambda)}{\partial a_{ij}} \frac{a_{ij}}{P([x]^{(l)}|\lambda)}}{\sum_{j=0}^{n} \sum_{l=1}^{L} \frac{\partial P([x]^{(l)}|\lambda)}{\partial a_{ij}} \frac{a_{ij}}{P([x]^{(l)}|\lambda)}}. \end{aligned} \tag{6}$$

Rewriting the numerator:

$$\begin{aligned} \sum_{l=1}^{L} \frac{\partial P([x]^{(l)}|\lambda)}{\partial a_{ij}} \frac{a_{ij}}{P([x]^{(l)}|\lambda)} &= \sum_{l=1}^{L} \frac{1}{m^{(l)}} \sum_{k=0}^{m^{(l)}-1} \frac{\partial P(x^{(l)}|\lambda, k)}{\partial a_{ij}} \frac{a_{ij}}{\frac{1}{m^{(l)}} \sum_{k=0}^{m^{(l)}-1} P(x^{(l)}|\lambda, k)} \\ &= \sum_{l=1}^{L} \sum_{k=0}^{m^{(l)}-1} \frac{1}{\sum_{k=0}^{m^{(l)}-1} P(\rho^k(x^{(l)})|\lambda)} \frac{a_{ij} \partial P(\rho^k(x^{(l)})|\lambda)}{\partial a_{ij}} \\ &= \sum_{l=1}^{L} \sum_{k=0}^{m^{(l)}-1} \frac{P(\rho^k(x^{(l)})|\lambda)}{\sum_{k=0}^{m^{(l)}-1} P(\rho^k(x^{(l)})|\lambda)} \left[\frac{\partial P(\rho^k(x^{(l)})|\lambda)}{\partial a_{ij}} \frac{a_{ij}}{P(\rho^k(x^{(l)})|\lambda)} \right]. \end{aligned}$$

Computing $\partial P(\rho^k(x^{(l)})|\lambda)/\partial a_{ij}$ [24]:

$$\begin{aligned} \frac{\partial P(\rho^k(x^{(l)})|\lambda)}{\partial a_{ij}} &= \frac{\partial}{\partial a_{ij}} \left(\sum_{i=0}^{n} \sum_{j=0}^{n} \alpha_t^{l_k}(i) a_{ij} b_j(\rho^k(x_{t+1}^{(l)})) \beta_{t+1}^{l_k}(j) \right) \\ &= \sum_{t=0}^{m^{(l)}-1} \alpha_t^{l_k}(i) b_j(\rho^k(x_{t+1}^{(l)})) \beta_{t+1}^{l_k}(j), \end{aligned}$$

we conclude that

$$\begin{aligned}\bar{a}_{ij} &= \frac{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\text{Expected number of transitions from } S_i \text{ to } S_j \text{ with } \rho^k(x^{(l)})}{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\text{Expected number of transitions from } S_i \text{ with } \rho^k(x^{(l)})} \\ &= \frac{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\frac{1}{\sum_{k=0}^{m^{(l)}-1}P(\rho^k(x^{(l)})|\lambda)}\sum_{t=0}^{m^{(l)}-1}\alpha_t^{l_k}(i)a_{ij}b_j(\rho^k(x_{t+1}^{(l)}))\beta_{t+1}^{l_k}(j)}{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\frac{1}{\sum_{k=0}^{m^{(l)}-1}P(\rho^k(x^{(l)})|\lambda)}\sum_{t=0}^{m^{(l)}-1}\alpha_t^{l_k}(i)\beta_{t+1}^{l_k}(j)},\end{aligned} \tag{7}$$

where $\alpha_t^{l_k}(i)$ and $\beta_t^{l_k}(j)$ are $\alpha_t(i)$ and $\beta_t(j)$ for $\rho^k(x^{(l)})$, respectively.

Following a similar reasoning with $b_i(v_j)$ and knowing that $\sum_{j=0}^{w} b_i(v_j) = 1$, for $1 \leq i \leq n$ and that (5) is a polynomial with respect to B, we arrive to

$$\begin{aligned}\bar{b}_i(v_j) &= \frac{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\text{Expected number of times in } S_i \text{ and observing } v_j \text{ with } \rho^k(x^{(l)})}{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\text{Expected number of times in } S_i \text{ with } \rho^k(x^{(l)})} \\ &= \frac{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\frac{1}{\sum_{k=0}^{m^{(l)}-1}P(\rho^k(x^{(l)})|\lambda)}\sum_{\substack{t=1 \\ \forall\rho^k(x_t^{(l)})=v_j}}^{m^{(l)}-1}\alpha_t^{l_k}(i)\beta_t^{l_k}(i)}{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\frac{1}{\sum_{k=0}^{m^{(l)}-1}P(\rho^k(x^{(l)})|\lambda)}\sum_{t=1}^{m^{(l)}-1}\alpha_t^{l_k}(i)\beta_t^{l_k}(i)}.\end{aligned} \tag{8}$$

In a similar way, for the continuous case:

$$\bar{\mu}_i = \frac{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\frac{1}{\sum_{k=0}^{m^{(l)}-1}P(\rho^k(x^{(l)})|\lambda)}\sum_{t=1}^{m^{(l)}-1}\alpha_t^{l_k}(i)\beta_t^{l_k}(i)\rho^k(x_t^{(l)})}{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\frac{1}{\sum_{k=0}^{m^{(l)}-1}P(\rho^k(x^{(l)})|\lambda)}\sum_{t=1}^{m^{(l)}-1}\alpha_t^{l_k}(i)\beta_t^{l_k}(i)}. \tag{9}$$

$$\bar{\sigma}_i = \frac{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\frac{1}{\sum_{k=0}^{m^{(l)}-1}P(\rho^k(x^{(l)})|\lambda)}\sum_{t=1}^{m^{(l)}-1}\alpha_t^{l_k}(i)\beta_t^{l_k}(i)(\rho^k(x_t^{(l)})-\mu_i)^2}{\sum_{l=1}^{L}\sum_{k=0}^{m^{(l)}-1}\frac{1}{\sum_{k=0}^{m^{(l)}-1}P(\rho^k(x^{(l)})|\lambda)}\sum_{t=1}^{m^{(l)}-1}\alpha_t^{l_k}(i)\beta_t^{l_k}(i)}. \tag{10}$$

We are in conditions to present the iterative procedure, the training algorithm for cyclic strings using Baum-Welch. It is described in Figure 6. The computational cost for each iteration is $O(Ln^2m^2)$.

Analogously, a cyclic training with Viterbi can be performed with the optimal sequence of states, $\hat{Q}$, of the cyclic strings, with the following reestimation formulae:

$$\hat{a}_{ij} = \frac{\sum_{l=1}^{L}\text{Number of transitions from } S_i \text{ to } S_j \text{ in } \hat{Q} \text{ with } \rho^k(x^{(l)})}{\sum_{l=1}^{L}\text{Number of transitions from } S_i \text{ in } \hat{Q} \text{ with } \rho^k(x^{(l)})}, \tag{11}$$

$$\hat{b}_i(v_j) = \frac{\sum_{l=1}^{L}\text{Number of times in } S_i \text{ and observing } v_j \text{ in } \hat{Q} \text{ with } \rho^k(x^{(l)})}{\sum_{l=1}^{L}\text{Number of times in } S_i \text{ in } \hat{Q} \text{ with } \rho^k(x^{(l)})}. \tag{12}$$

Figure 6: Training algorithm for cyclic strings using Baum-Welch.

Input: λ: model, $X = \{[x]^{(1)}, [x]^{(2)}, \ldots, [x]^{(L)}\}$: set of cyclic strings

Output: $\bar{\lambda}$: trained model

var M_α, M_β, A, B: matrix $[0\,..\,n][1\,..\,m]$ of $\mathbb{R}$ // supossing that all the strings // have the same size m

begin
- $\bar{\lambda} = \lambda$
- **while** there is no convergence **do**
  - **for** l **in** $1\,..\,L$ **do**
    - $P_{total} = 0$
    - **for** k **in** $0\,..\,m^{(l)} - 1$ **do**
      - $P_{total} += \texttt{Forward}\ (\bar{\lambda},\ \rho^k(x^{(l)}),\ M_\alpha)$
    - **for** k **in** $0\,..\,m^{(l)} - 1$ **do**
      - $P = \texttt{Forward}\ (\bar{\lambda},\ \rho^k(x^{(l)}),\ M_\alpha)$
      - $P = \texttt{Backward}\ (\bar{\lambda},\ \rho^k(x^{(l)}),\ M_\beta)$
      - Compute expected values using (7) and (8), and store them in A and B
  - Reestimate $\bar{\lambda}$ using (7) and (8)
- **return** $\bar{\lambda}$

end

function `Forward`(λ: model, x: string, **output** M_α: matrix $[0\,..\,n][1\,..\,m]$ of $\mathbb{R}$): $\mathbb{R}$

begin
- Iterative computation of $P = \sum_{i=0}^{n} \alpha_m(i)$ for x, in matrix M_α
- **return** P

end

function `Backward`(λ: model, x: string, **output** M_β: matrix $[0\,..\,n][1\,..\,m]$ of $\mathbb{R}$): $\mathbb{R}$

begin
- Iterative computation of $P = \sum_{i=0}^{n} \beta_0(i)$ for x, in matrix M_β
- **return** P

end

Figure 7: Training algorithm for cyclic strings using Viterbi.

Input: λ: model, $X = \{[x]^{(1)}, [x]^{(2)}, \ldots, [x]^{(L)}\}$: set of cyclic strings

Output: $\hat{\lambda}$: trained model

var

$\hat{Q}, Q'$: vector $[1\,..\,m]$ of $\mathbb{N}$ // supossing that all the strings

A, B: matrix $[0\,..\,n][1\,..\,m]$ of $\mathbb{R}$ // have the same size m

begin

- $\hat{\lambda} = \lambda$
- **while** there is no convergence **do**
  - **for** l **in** $1\,..\,L$ **do**
    - $\hat{P} = -\infty$
    - **for** k **in** $0\,..\,m^{(l)} - 1$ **do**
      - $P' =$ `Viterbi` $(\hat{\lambda},\ \rho^k(x^{(l)}),\ Q')$
      - **if** $P' > \hat{P}$ **then**
        - $\hat{P} = P'$
        - $\hat{Q} = Q'$
    - Compute frequencies using (11) and (12) with $\hat{Q}$, and store them in A and B
  - Reestimate $\hat{\lambda}$ using (11) and (12)
- **return** $\hat{\lambda}$

end

function `Viterbi`(λ: model, x: string, $\hat{Q}$: vector $[1\,..\,m]$ *of* $\mathbb{N}$): $\mathbb{R}$

begin

- Iterative computation of $\hat{P} = \max_{i=0}^{n} \phi_m(i)$ and $\hat{Q}$ for x
- **return** $\hat{P}$

end

The iterative training algorithm with Viterbi and cyclic strings is shown in Figure 7.

Following the line of thought in [25]:

Theorem 1. *The Viterbi training for cyclic strings converges in Zangwill's global convergence sense [25, 26].*

Proof: What needs to be shown is that $P([x]|\lambda)$ is an ascent function for this algorithm. Let Q^* and $\hat{Q}$ be two optimal sequences of states such that, $Q^* = \text{argmax}_Q\, P([x], Q|\lambda)$ and $\hat{Q} = \text{argmax}_Q\, P([x], Q|\hat{\lambda})$, then:

$$\begin{aligned} \max_Q P([x], Q|\hat{\lambda}) &\geq P([x], Q^*|\hat{\lambda}) \\ &= \max_{\lambda'} P([x], Q^*|\lambda') \\ &= \max_{\lambda'} \Big(\max_Q \big(\max_r P(\rho^r(x), Q|\lambda') \big) \Big) \\ &\geq \max_Q P([x], Q|\lambda). \end{aligned} \tag{13}$$

The maximization over λ' in (13) can be replaced by the cyclic Viterbi training explained above. □

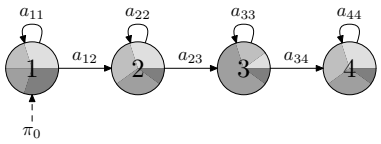
Note that the cyclic Baum-Welch training needs a good initialization. The same happens with the cyclic Viterbi training. In Section 6 we propose a heuristic to solve this.

5. Cyclic Linear HMMs

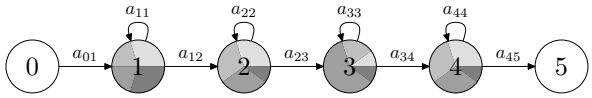
In Section 1 we concluded that left-to-right topologies are the best suited for modelling contours. We can go further and say that the linear topology (see Figure 2b) is possibly the best one in this context, because if we use a Bakis topology (see Figure 2c) or one with more transitions per state, complexity increases, as it happens with ergodic topologies. Moreover, if we want to model cyclic strings, as in our case, linear models have interesting capabilities, that we will explore in the following.

For this, we will use an alternative definition of Markov models that was popularized by the *Hidden Markov Model Toolkit* (HTK) [27]. In it, in addition to the initial non-emitting state, there is a final non-emitting state[1]. In Figure 8 there is an example. In Figure 9 an example of the

[1]This new state slightly varies the algorithms discussed so far. We will not show these variations because they are trivial.



(a)



(b)

Figure 8: (a) A linear HMM. (b) A linear HMM using the topology with two non-emitting states, the initial and the final ones.

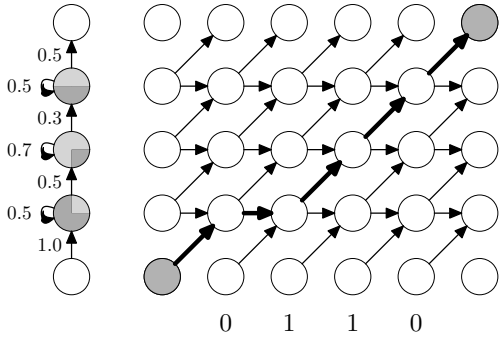


Figure 9: Graph for a linear HMM and a string of size 4. The optimal alignment is shown with thicker arrows.

iterative computation of the Viterbi score with this topology is shown. As we can see, there is a resemblance between this graph and the alignment graph of the Cyclic DTW [6], and then, the computational cost of this particular case is $O(nm)$ and not $O(n^2m)$. From now on, and as with the Cyclic DTW, we will call optimal alignment to the optimal sequence of states that produces the Viterbi score. In it the alignment is produced between a state and a segment of contiguous elements along the contour.

To properly model cyclic strings, HMMs should take into account that any symbol of the string can be emitted by the first emitting state and when such a symbol has been chosen as emitted by this state, its previous symbol must be emitted by the last state. Thus, we can use Linear HMMs in a way similar to cyclic strings. A Cyclic Linear HMM (CLHMM) can be seen as the set obtained by cyclically shifting a conventional Linear HMM (LHMM).

Definition 3. *Let $\lambda = (A, B)$ be an LHMM. Let $\rho(A)$ be the following transformation:*

$$A = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & a_{11} & a_{12} & 0 & \dots & 0 \\ 0 & 0 & a_{22} & a_{23} & 0 & \dots & 0 \\ 0 & \dots & 0 & \ddots & \ddots & 0 & 0 \\ 0 & \dots & \dots & a_{nn} & a_{nn+1} & 0 \\ 0 & \dots & \dots & 0 & 0 \end{bmatrix},$$

$$\rho(A) = \begin{bmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ 0 & a_{22} & a_{23} & 0 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & 0 & \dots & 0 \\ 0 & \dots & 0 & a_{nn} & a_{nn+1} & 0 & 0 \\ 0 & \dots & \dots & a_{11} & a_{12} & 0 \\ 0 & \dots & \dots & 0 & 0 \end{bmatrix}.$$

Let $\rho(B)$ be $\rho(b_1 \dots b_n) = b_2 \dots b_n b_1$ (where b_i are rows from matrix B). The composition of r cyclic shifts of λ is defined as $\rho^r(\lambda) = (\rho^r(A), \rho^r(B))$. Two LHMMs λ and λ' are cyclically equivalent if $\lambda = \rho^r(\lambda')$, for some r. The equivalence class of λ is $[\lambda] = \{\rho^r(\lambda) : 0 \leq r < n\}$ and it is called a Cyclic LHMM. *Any of its members is a representative of the CLHMM.*

In Figure 10, there is an example of a CLHMM.

We can also generalize the notion of Viterbi scores.

Definition 4. *The Viterbi score for a cyclic string $[x_1 x_2 \dots x_m]$ given a CLHMM $[\lambda]$ is defined as*

$$\hat{P}([x]|[\lambda]) = \max_{0 \leq r < n} \big(\max_{0 \leq s < m} \hat{P}(\rho^s(x)|\rho^r(\lambda)) \big),$$

and this score has associated an optimal alignment.

This is computationally expensive, but the following lemma shows that in order to compute the Viterbi score for a cyclic string and a CLHMM, one can simply choose a representative of the CLHMM and compute the Viterbi score between it and the cyclic string.

Lemma 1. $\hat{P}([x]||[\lambda]) = \hat{P}([x]|\lambda) = \max_{0 \leq s < m} \hat{P}(\rho^s(x)|\lambda)$.

Proof: Consider an optimal alignment $\hat{Q}_1$ that represents a maximum probability between λ and $\rho^{s_1}(x)$, for some s_1, then, there is an optimal alignment $\hat{Q}_2$ between $\rho(\lambda)$ and $\rho^{s_2}(x)$, for some s_2, such that $\hat{Q}_2$ exactly represents the same emitted symbols for each state as $\hat{Q}_1$. $\square$

Therefore, the optimal alignment can be computed by means of the conventional Viterbi score on m conventional strings in $O(m^2 n)$ time.

We propose a more efficient algorithm to evaluate the Viterbi score. The method computes the optimal alignment that begins in any state, visits all the states and does not visit again any state once it has left it. The algorithm is inspired by the Maes' algorithm for the Cyclic Edit Distance (CED) [22] and computes the Viterbi score in $O(mn \log m)$ time. The score is computed on an extended graph where the original string appears concatenated with itself in the horizontal axis and alignments must begin and end in nodes with the same colour (corresponding to the size of the string) (see Figure 11). The efficiency of the algorithm is based on the "non-crossing paths" property [22]: Let $\hat{Q}_i$ be the optimal alignment beginning at node $(i, 0)$ and ending at node $(m + i + 1, n + 1)$ in the extended graph and let j, k, and l be three integers such that $0 \leq j < k < l \leq m$; there is an optimal path starting at node $(k, 0)$ and arriving to $(k+m+1, n+1)$ that lies between $\hat{Q}_j$ and $\hat{Q}_l$.

This property leads to a divide and conquer procedure: when $\hat{Q}_j$ and $\hat{Q}_l$ are known, $\hat{Q}_{(j+l)/2}$ is computed by only taking into account those nodes of the extended graph lying between $\hat{Q}_j$ and $\hat{Q}_l$; then, optimal alignments bounded by $\hat{Q}_j$ and $\hat{Q}_{(j+l)/2}$ and optimal alignments bounded by $\hat{Q}_{(j+l)/2}$ and $\hat{Q}_l$ can be recursively computed. The recursive procedure starts after computing $\hat{Q}_0$ (by means of a standard Viterbi computation) and $\hat{Q}_m$, which is $\hat{Q}_0$ shifted m positions to the right. Each recursive call generates up to two more recursive calls and all the calls at the same recursion depth amount to $O(mn)$ time; therefore, the algorithm runs in $O(mn \log m)$ time. The algorithm is shown in Figure 12.

In principle, we could adopt a symmetric approach defining a cyclic shift on the states of the Linear HMMs to obtain the same Viterbi score. This is appealing because usually $n < m$ and, therefore, "doubling" the HMM in the extended graph instead of the string would lead to an $O(mn \log n)$ algorithm. This would be better than $O(mn \log m)$. However, it cannot be directly

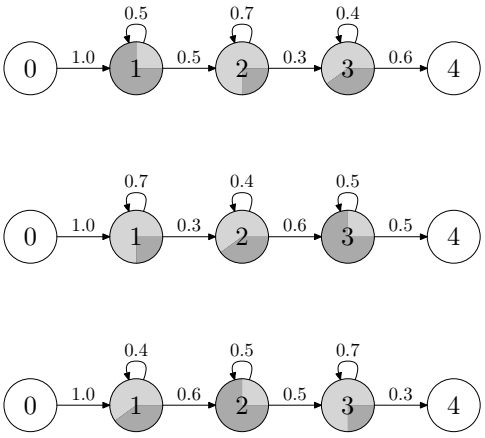


Figure 10: A CLHMM represented by its set of LHMMs.

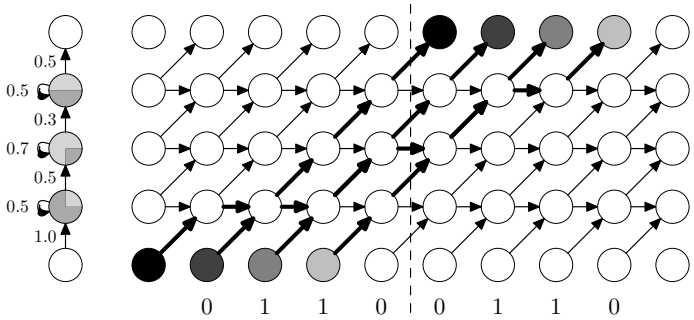


Figure 11: Extended graph for a Linear HMM and a cyclic string of size 4. The optimal alignments for each starting point are shown with thicker arrows, one of them is the optimal cyclic alignment (the one with the highest score).

Figure 12: Divide-and-conquer algorithm for computing $P([x]|\lambda)$.

Input: x: string, λ: model

Output: $\hat{p} : \mathbb{R}$

var $\hat{Q}$: vector $[0..m]$ alignment paths

begin

$p^* = \hat{P}(\rho^0(x)|\lambda)$

Let $\hat{Q}[0]$ be the optimal path of the alignment in the previous calculation

Let $\hat{Q}[m]$ be equal to $\hat{Q}[0]$ but moved m nodes to the right

if $m > 1$ **then**

$\hat{p} =$ `min(`$\hat{p}$`, NextStep(`$x \cdot x$`, `y`, 0, `m`))`

return $\hat{p}$

end

function `NextStep(`X*:* string, λ*:* model, $l : \mathbb{N}$, $r : \mathbb{N}$`)`:$\mathbb{R}$

begin

$c = l + \lceil \frac{r-l}{2} \rceil$

$p = \hat{P}(X_{c:c+m}, \lambda)$ with $\hat{Q}[l]$ and $\hat{Q}[r]$ known

if $l + 1 < c$ **then**

$p =$ `min(`p`, NextStep(`X`, `y`, `l`, `c`))`

if $c + 1 < r$ **then**

$p =$ `min(`p`, NextStep(`X`, `y`, `c`, `r`))`

return p

end

done:

Lemma 2. $P([x]|[\lambda]) \neq \max_{0\leq r<n} P(x|\rho^r(\lambda))$.

Proof: We show a counterexample. Let $[x] = v_1v_2v_1$ be a cyclic string on the alphabet $\Sigma = \{v_1, v_2\}$. Let $[\lambda]$ be a CLHMM with two emitting states and $a_{01} = 1$, $a_{11} = 0.5$, $a_{12} = 0.5$, $a_{22} = 0.5$, $a_{23} = 0.5$, $b_{01} = 1$, and $b_{12} = 1$. The definition of the Viterbi score in Lemma 1 leads to a value of 0.125 (for the string $\rho^2(v_1v_2v_1) = v_1v_1v_2$). If we try to perform a cyclic shift in the Linear HMM, we have two possible cyclic shifts, both give 0 as the Viterbi score. □

All is not lost. We can introduce a modification on the HMM using the following definition.

Definition 5. *Let $[\lambda] = (A, B)$ be a CLHMM. $\iota(\lambda)$ is the operation that performs a cyclic shift ($\rho(\lambda)$) and inserts a copy of the first emitting state before the last state, but its transition to the next state has the value of its self transition.*

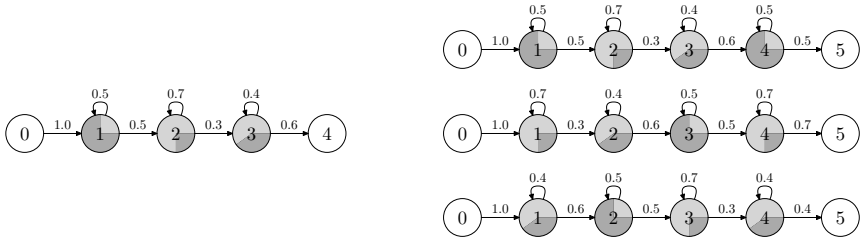
Note that the result of $\iota(\lambda)$ is not a valid HMM, since the transitions from the state that is inserted do not sum one (see Figure 13). However, we will show that the corresponding probabilities are not changed.

Let $[\lambda] = (A, B)$ be a CLHMM, let $[x]$ be a cyclic string. Then,

Theorem 2.

$$\hat{P}([x]|[\lambda]) = \max_{0\leq r<n} \Big(\max \big(\hat{P}(x|\rho^r(\lambda)), \hat{P}(x|\iota^r(\lambda)) \big) \Big).$$

Proof: Each alignment induces a segmentation on x. All the symbols in a segment are aligned with the same state of the CLHMM. There is a problem when $x_{m-p}x_{m-p+1}\ldots x_m$ and $x_1x_2\ldots x_q$, for some p, $q \geq 0$, belong to the same segment of x. In that case, the optimal alignment cannot be obtained by simply cyclic shifting λ, since x_m must be aligned with the state n and x_1 must be aligned with the state 1, i.e., they never fall in the same segment. The model $\iota^r(\lambda)$, formed by inserting to $\rho^r(\lambda)$ the first emitting state after the last one, permits to align $x_{m-p}x_{m-p+1}\ldots x_m$ and $x_1x_2\ldots x_q$ with the first state, since this state also appears at the end of $\iota^r(\lambda)$. On the other hand, there is another problem: let us suppose we have now the complete segment at the beginning of the string, $p+q$ symbols, then the first self transition must be executed $p+q-1$ times, but if the



(a) (b)

Figure 13: (a) A CLHMM $[\lambda]$ represented by a representative (an LHMM). (b) The corresponding LHMMs for the $\iota^r(\lambda)$ operation, for $0 \leq r < n$ (where $n = 3$). From top to bottom, $\iota^0(\lambda)$, $\iota^1(\lambda)$ and $\iota^2(\lambda)$

segment is in the situation explained above, the first self transition will be executed just $p + q - 2$ times. The transition to the last non-emitting state provides this necessary extra transition. □

Corollary 1. *For each value of* r, $\hat{P}(x|\rho^r(\lambda))$ *can be obtained as a subproduct of the computation of* $P(x|\iota^r(\lambda))$.

Proof: The graph underlying $P(x|\rho^0(\lambda))$ is a subgraph of the one underlying $P(x|\iota^0(\lambda))$. The value of $P(x|\rho^r(\lambda))$ and $P(x|\iota^r(\lambda))$, for each r, can be obtained by computing optimal alignments in an *extended graph* similar to the one in Figure 11, but now "doubling" the LHMM. The value of $P(x|\rho^r(\lambda))$ and $P(x|\iota^r(\lambda))$, for each r, can be obtained by computing optimal alignments in an *extended graph* similar to the one in Figure 11, but now "doubling" the LHMM. It should be taken into account that, unlike in Maes' algorithm, the optimal path starting at $(r, 0)$ can finish either at node $(r + n - 1, m)$ or $(r + n, m)$ and the recursive computation can be applied just using the optimal alignments between $\rho^r(\lambda)$ and x as a new left or right border. □

In Figure 14 there is an example of this.

Finally, note that the proposed divide-and-conquer algorithm can be used, obviously, to speed up recognition and training with Viterbi and cyclic strings. Unfortunately, this cannot be extended to the *forward* or *backward* procedures, because there are no optimal alignments in the graph, and then, we cannot use the non-crossing property.

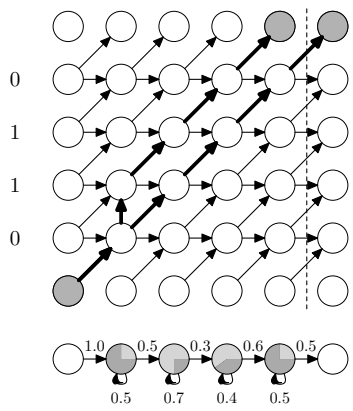


Figure 14: The same example of graph that appears in Figure 9, but in this case, the LHMM appears bellow with the operation $\iota^0(\lambda)$. We can see that $\hat{P}(x|\rho^0(\lambda))$ can be easily computed with this graph.

6. A Better Heuristic for Selecting the Starting Point

We will see in the experiments that the labelling of the training samples gives us the following heuristic for obtaining a starting point that improves those described in Section 3.3.

We need to perform a preprocessing. For it we use cyclic DTW (CDTW) [6] that, apart from returning the cost (distance) of the cyclic alignment, can also return the corresponding cyclic shift of one of the strings for the alignment with the other string (see Figure 16). Starting from a set of training samples, our aim is to choose an appropriate starting point for them. We select a representative (the centroid of the class using CDTW) and an arbitrary starting point for it. With the representative of each class and its starting point, we compute the CDTW for each one of the other members of the class and the representative, obtaining the cyclic shift of the alignment that defines a good starting point for each of them. The preprocessing procedure is shown in Figure 15. Once we have an appropriate starting point for the training samples, we can train the model of each class as if the cyclic strings were ordinary strings.

In a similar way, to classify a new sample, we begin by finding adequate starting points for it (one for each class). These starting points are computed by CDTW with the representative of each class. Thus, with this starting point for each class we can compute probabilities (or Viterbi scores) in a conventional way.

Although, as we will see in the section of experiments (Section 8), this solution has worse results than the methods that we present in previous sections, both training and recognition are much faster. Moreover, as we mentioned before, this training can also be used as a good initialization

Figure 15: Preprocessing algorithm.

Input: X: set of training strings

Output: X: set of training strings with a new cyclic shift

begin

  for $x \in X$ **do**

    d, $shift = CDTWS(x, \texttt{Representative}(x))$

    $x = \rho^{\text{shift}}(x)$

  return X

end

for the training methods of Section 4.

7. Some Considerations on Computational Complexity

In the next section we experimentally show the performance of our proposals in terms of classification rates. Here we analyse their computational time complexity. Table 1 shows the computational time complexities of the different approaches.

In Section 5 we mentioned that, amongst left-to-right topologies, the linear topology seems to be the best for modelling strings. In Section 8.2 we experimentally prove it in our context. Then, considering that our topology is going to be linear, we do not have to take into account every possible connection between the states. Then, the computational cost is considerably reduced for all the approaches (obviously, with the exception of the ergodic model). The circular topology is also a linear topology where the last emitting state is connected to the first one. That does not increase the computational cost.

The lowest computational cost for classification corresponds to Fourier descriptors, circular topology and our heuristic. We can classify in $O(mn)$ time. In the cyclic approach, as we use Viterbi scores, both in the cyclic Viterbi and in the cyclic Baum-Welch we can classify in $O(mn \log n)$.

The computational time complexity of training is expressed for each iteration and considering that it is performed with a set of L strings. The computational cost of Fourier descriptors, circular topology and our heuristic is the lowest again. We can train in $O(Lmn)$ time for each iteration. In

Figure 16: CDTWS: Cyclic DTW algorithm that also returns the cyclic shift.

Input: x, y: strings

Output: d^*: $\mathbb{R}$, *shift*: $\mathbb{N}$ // Distance and cyclic shift

var P: vector $[0..m]$ alignment paths

begin

$d^* = \min(DTW(\rho^0(x), y), DTW(\rho^0(x)x_0, y))$

Let $P[0]$ be the optimal path of the alignment obtained in the previous calculation

Let $P[m]$ be equal to $P[0]$ but moved m nodes to the right

if $m > 1$ **then**

$d =$ `NextStep(`$x \cdot x$, y, 0, m, *shift*`)`

if $d^* < d$ **then**

$shift = 0$

else

$d^* = d$

return d^*, *shift*

end

function `NextStep(`X*:* string, y*:* string, $l : \mathbb{N}$, $r : \mathbb{N}$, *rshift:* $\mathbb{N}$`)`: $\mathbb{R}$

begin

$c = l + \lceil \frac{r-l}{2} \rceil$

$shift = c$

$d = \min(DTW(X_{c:c+m}, y), DTW(X_{c:c+m+1}, y))$ with $P[l]$ and $P[r]$ known

if $l + 1 < c$ **then**

$dl =$ `NextStep(`X, y, l, c, *shift*`)`

if $dl < d$ **then**

$d = dl$

$rshift = shift$

if $c + 1 < r$ **then**

$d =$ `min(`d, `NextStep(`X, y, c, r, *shift*`))`

if $dr < d$ **then**

$d = dr$

$rshift = shift$

return d

end

the case of the cyclic Viterbi, we can train in $O(Lmn \log n)$. In the case of the cyclic Baum-Welch, the computational complexity of training is higher, $O(Lmn^2)$, because it is not possible to use the non-crossing property (see Section 4).

Table 1: Computational time complexity for classification and training for: ergodic topology (Ergodic), Fourier descriptors (FDs), circular topology (Arica), our heuristic (Heuristic), cyclic Viterbi (CViterbi) and cyclic Baum-Welch (CBW).

Problem	**Method**			
	Ergodic	FDs/Arica/Heuristic	CViterbi	CBW
Classification	$O(mn^2)$	$O(mn)$	$O(mn \log n)$	$O(mn \log n)$
Training (iteration)	$O(Lmn^2)$	$O(Lmn)$	$O(Lmn \log n)$	$O(Lm^2n)$

8. Experiments

In order to assess the behaviour of the presented algorithms, we performed comparative experiments on a shape recognition task on publicly available databases:

- MPEG7 CE-Shape-1 corpus part B (MPEG7B). It contains 1400 shapes (see Figure 17) divided in 70 categories, each category with 20 images [28].
- Silhouette corpus [29]. It contains 1070 silhouettes (see Figure 18). The shapes belong to 41 categories representing different objects.
- He-Kundu corpus [8]. A corpus that contains 8 images (see Figure 19).
- Subset 1 corpus. A subset of the MPEG7B corpus. It contains 140 shapes (see Figure 20) divided in 7 categories, each category with 20 images [14].
- Corpus of airplane shapes. It contains 210 shapes (see Figure 21) divided in 7 categories, each category with 30 images [14].
- Corpus of shapes of vehicles. It contains 120 shapes (see Figure 22) divided in 4 categories, each category with 30 images [14].

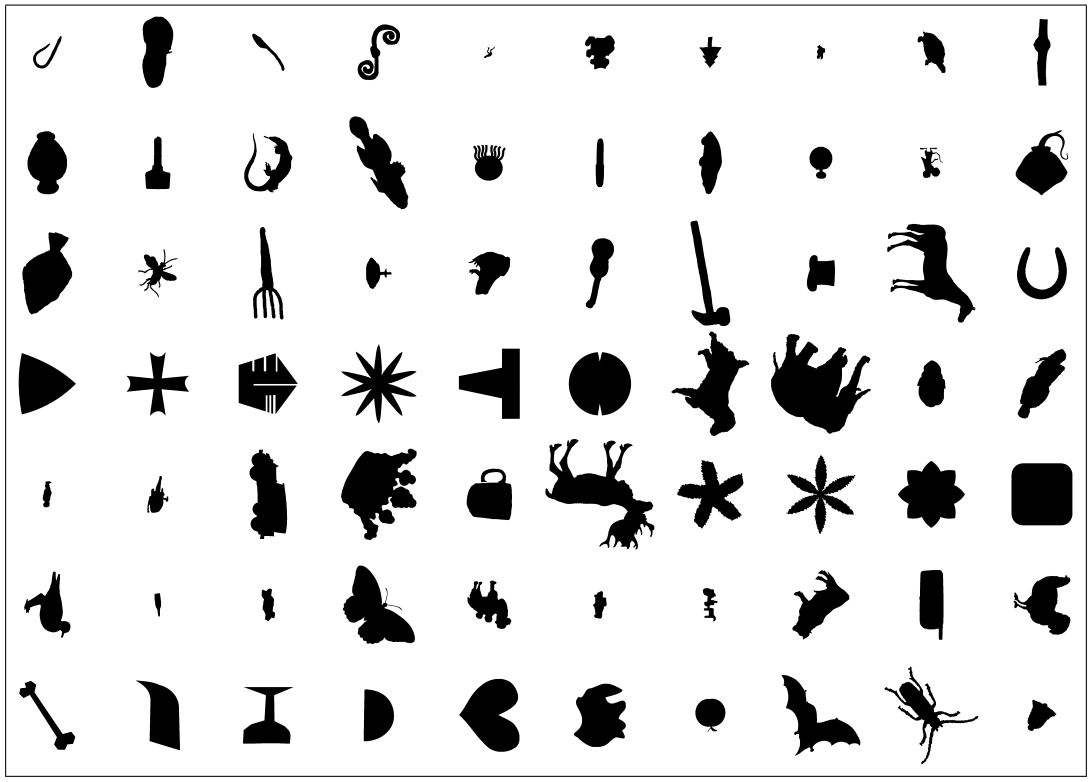


Figure 17: Some images in MPEG7 CE-Shape-1 Part B corpus. A sample for each class.

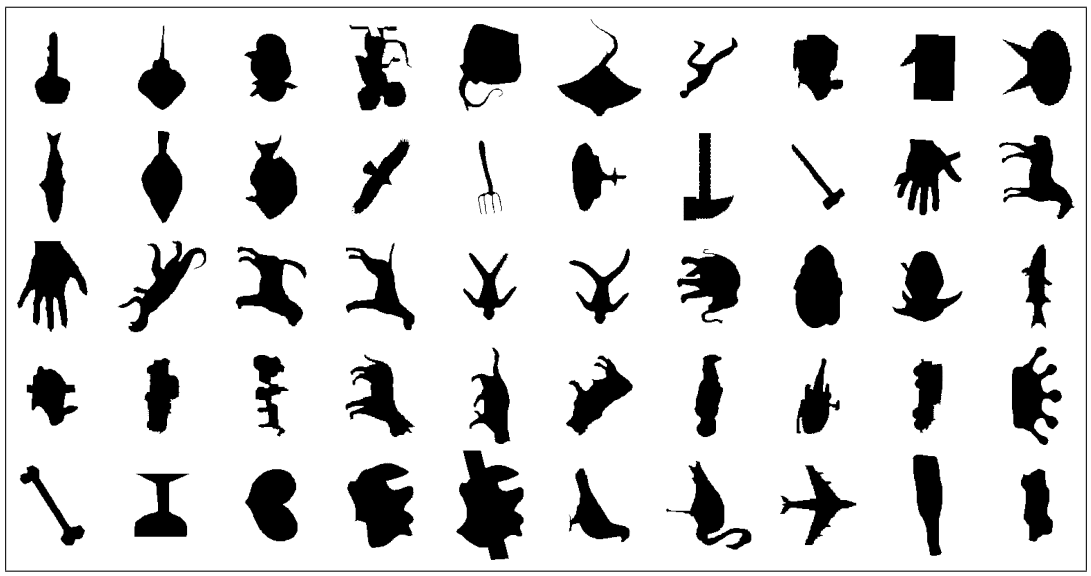


Figure 18: Some images in Silhouette corpus. A sample for each class.

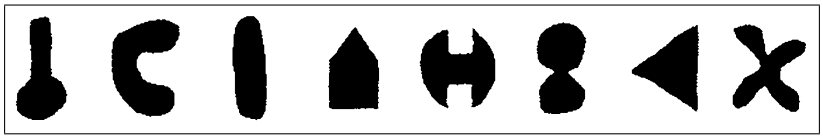


Figure 19: Images in He-Kundu corpus.

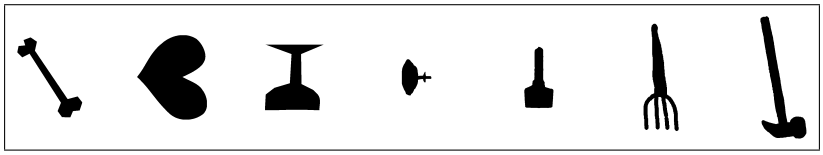


Figure 20: Some images in subset 1 corpus. A sample for each class.

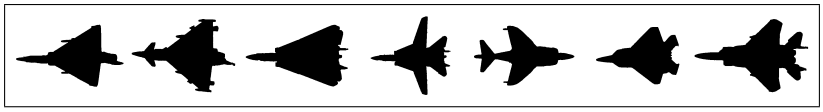


Figure 21: Some images in the corpus of airplane shapes. A sample for each class.

The outer contours of the images were extracted as sequences of points. A random starting point in the sequences was also selected.

In Sections 8.1, 8.2, 8.3 and 8.4, 128 landmark points were sampled uniformly. As it is customary in the literature of HMMs and shape recognition we used the curvature shape descriptor.

In Section 8.5, we have experiments with a multidimensional shape descriptor [5]. As in [5], 100 landmark points were sampled uniformly.

The evaluation was done with classification rates for different number of states (we train an HMM for each category): 10 to 120 in steps of 10. We use a gaussian per state.

All the experiments were performed using cross-validation [30] except the ones with subset 1 corpus that were performed with a leaving-one-out approach [30] for comparing with other results in the bibliography. For classification we use the Viterbi scores.

8.1. Invariance to the Starting Point

In Section 3 several solutions to the starting point invariance problem are commented. Here we compare these solutions with the heuristic (proposed in Section 6) and a linear left-to-right topology. In particular, we compare our proposal with the circular topology [10], the election of the starting point using Fourier descriptors [8], and the ergodic topology [12, 14].

In Figures 23a and 23b the results of the comparison are shown, for MPEG7B and Silhouette

Figure 22: Some images in the corpus of shapes of vehicles. A sample for each class.

corpora. The ergodic topology obtains the worst results[2]. The election of the starting point and the circular topology (especially the latter) happen to be the most competitive in front of our heuristic. Taking into account the simplicity of the heuristic proposed, it obtains very good results in comparison with them.

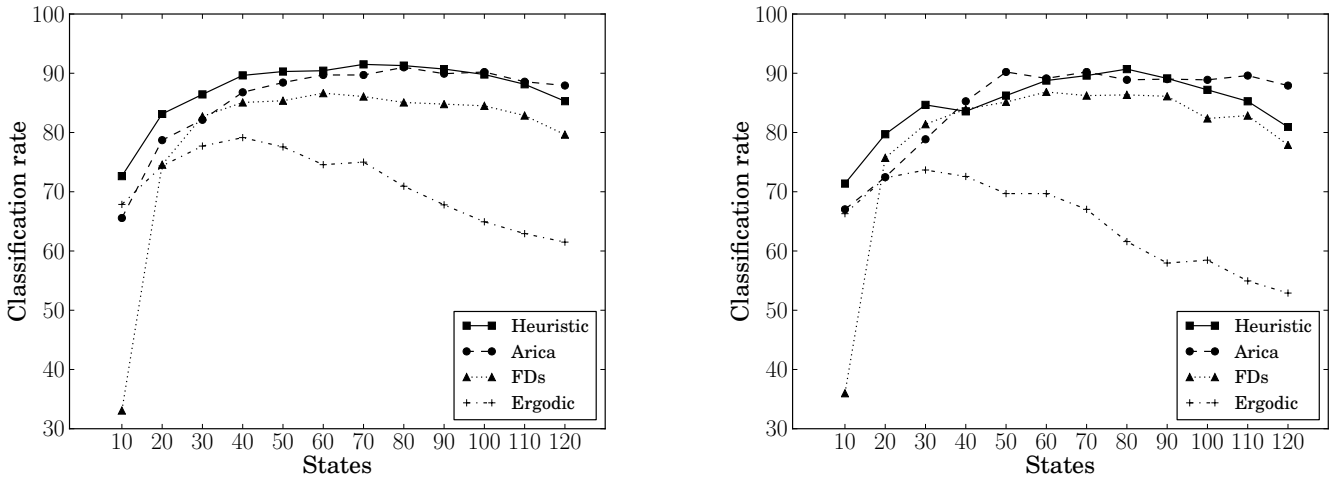


Figure 23: Classification rates for the comparison between the circular topology (Arica), the election of the starting point with Fourier descriptors (FDs), the ergodic topology (Ergodic) and our heuristic (Heuristic). With corpora (a) MPEG7B and (b) Silhouette.

8.2. Left-to-right Topologies

In Section 3 we mentioned that left-to-right topologies are the most suitable for modelling strings. In Section 5 we specify that, within these topologies, the linear topology seems to be the best one for this purpose, because having more transitions increases the complexity of the model. Here we empirically prove this affirmation with a comparison between three left-to-right topologies: linear, Bakis and the one with four transitions per state. The last one is similar to Bakis but with another transition to the next of the next of the next state. The method used for training and classifying is our heuristic.

The results are shown in Figures 24a and 24b. As we can see the linear topology outperforms the others.

[2]We will talk more about this topology in Section 8.4.

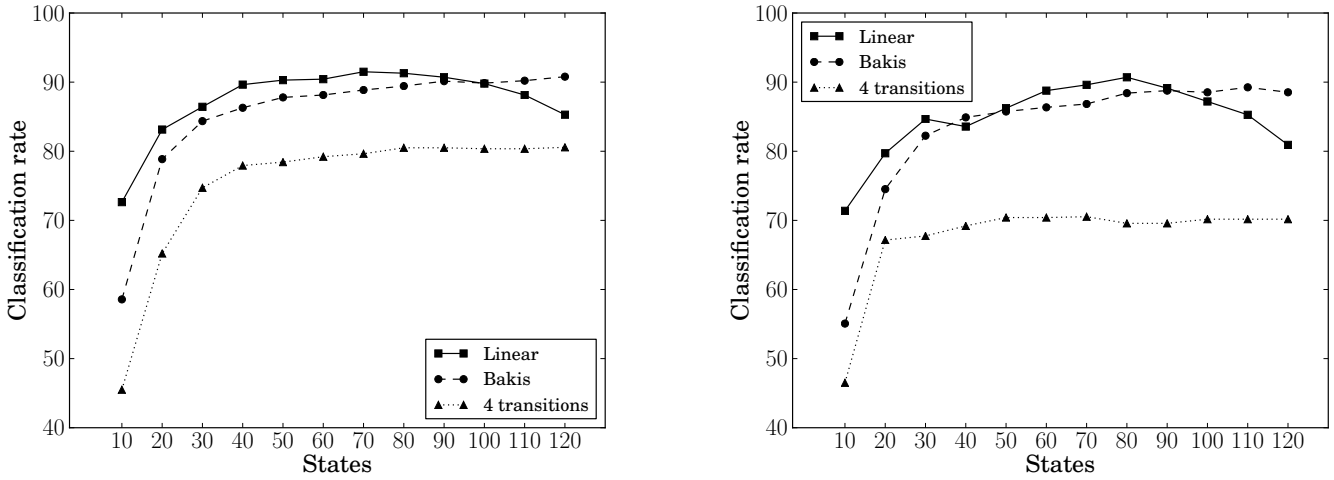


Figure 24: Classification rates for the comparison between different left-to-right topologies. Linear topology, Bakis topology and topology of four transitions. Our heuristic is used for training and classifying. With corpora (a) MPEG7B and (b) Silhouette.

8.3. Cyclic Approach

In this section, we compare our cyclic approach, Baum-Welch and Viterbi for cyclic strings (see Section 4) with our heuristic and the circular topology [10]. Cyclic training is initialized using our heuristic.

Comparative results are in Figures 25a and 25b. We can observe that cases where cyclic Baum-Welch and cyclic Viterbi win predominate. In Table 2, there is a summarise with the best results for each method.

Figure 26 shows the results for each class for the corpus MPEG7B. We can also see here that our cyclic approaches, both cyclic Viterbi and cyclic Baum-Welch, are more robust.

It is worth noting that although both techniques obtain similar results, cyclic Baum-Welch training has a higher computational cost. Taking into account that we have linear topologies, if we use cyclic Viterbi, we can train with the methods presented in Section 5 and then, achieve a $O(Lmn \log n)$ computational cost for each iteration.

8.4. More about the Ergodic Topology

In Section 8.1 we experimentally saw that the ergodic topology does not offer good results. However, in the literature there are works [8, 12–14] where this topology is used.

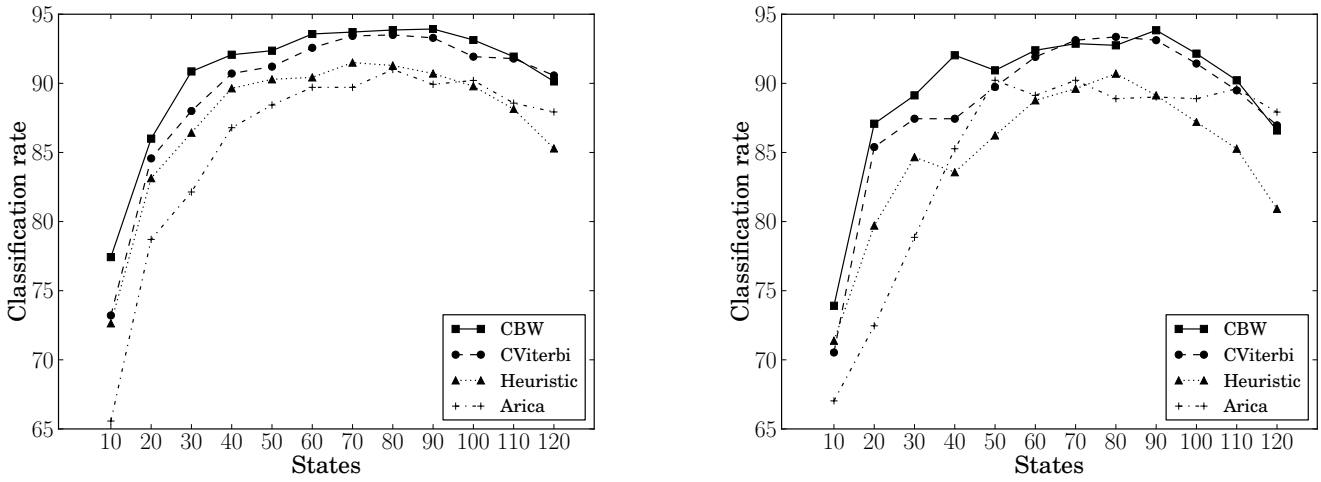
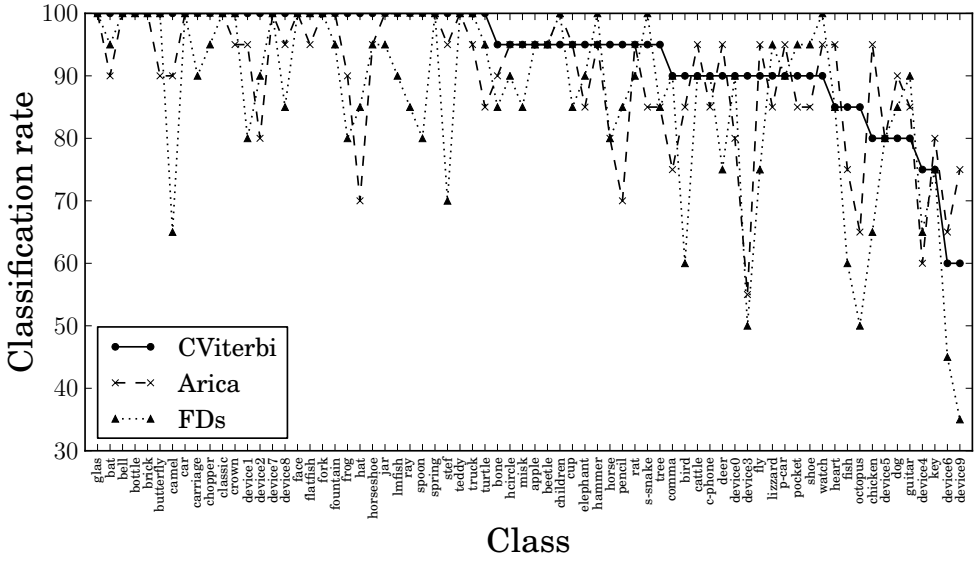


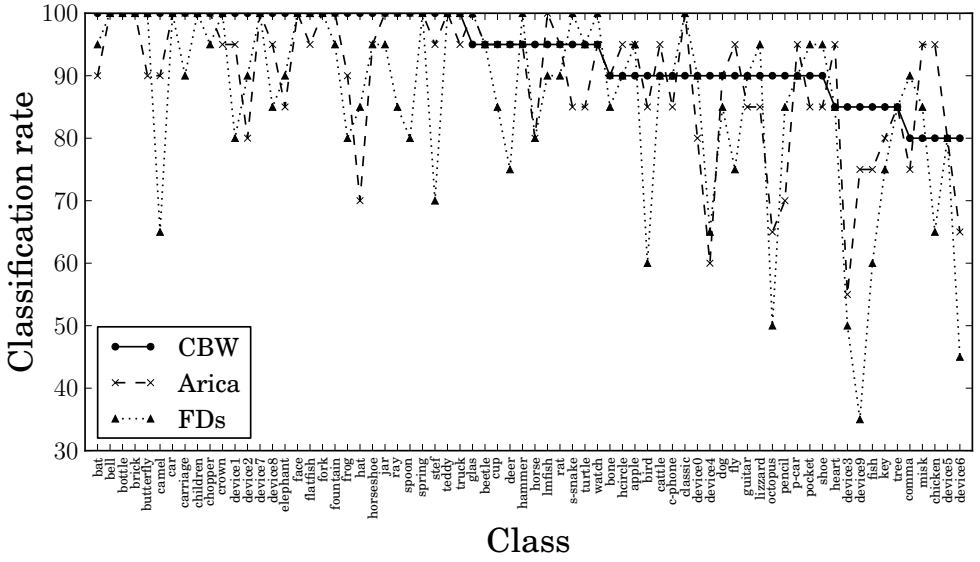
Figure 25: Classification rates for the comparison between: cyclic Baum-Welch (CBW), cyclic Viterbi (CViterbi), our heuristic (Heuristic) and circular topology (Arica). With corpora (a) MPEG7B and (b) Silhouette.

Table 2: Classification rates of the best results (see Figure 25) for the comparison between: cyclic Baum-Welch (CBW), cyclic Viterbi (CViterbi), our heuristic (Heuristic) and circular topology (Arica). With corpora (a) MPEG7B and (b) Silhouette. Bold entries show the best results in the comparison.

Corpus	**Method**			
	Arica	Heuristic	CViterbi	CBW
MPEG7B	89.93	91.50	93.50	**93.93**
Silhouette	90.22	90.70	93.36	**93.84**



(a)



(b)

Figure 26: Classification rates for all classes from MPEG7B (classes are sorted by the results of the first method) for the best results of each method (see Table 2). (a) Comparison between cyclic Viterbi (CViterbi), circular topology (Arica) and Fourier descriptors (FDs). (b) Comparison between cyclic Baum-Welch (CBW), circular topology (Arica) and Fourier descriptors (FDs).

More specifically, in [12] experiments are performed with this topology. For training, the authors choose a number of states with BIC (*Bayesian Inference Criterion*) [31] over a clustering of curvatures. The obtained results are good enough but their corpora have few samples and classes. They use a subset of the MPEG7B corpus of 6 classes with 10 samples per class (a subset of subset 1). They also use He-Kundu corpus for performing an experiment of invariance to the starting point achieving a classification rate of 100%. This way, they conclude that HMMs with an ergodic topology are enough for obtaining this invariance. In our opinion, this experiment is not enough for claiming that affirmation. For this corpus we also achieve a 100% with the cyclic Viterbi for training and classifying.

In [13], a work of the same authors, another subset of MPEG7B is used (a subset of subset 1, with 12 samples per class). We will call this corpus subset 2, as it is done in [14]. In this case, they use a reference rotation for the election of the starting point. Instead of using BIC for obtaining the number of states, they use a fixed number of states. In [14], the authors, parting from the work of [12, 13], try to improve their results with a training based on GPD (*Generalized probabilistic descent method*). They also use the subset 2 and create a new one, subset 1. With subset 2 they obtain a classification rate of 97.63% (the best result with this subset in [13] is 98.8%). With subset 1 they obtain a 96.43% (in [13] there are no results with this subset). With subset 1 and cyclic Viterbi or cyclic Baum-Welch, we achieve a 99.29% (see Figure 27 and Table 3), that even outperforms the classification rate of [13] with subset 2. None of the previous works show results with the entire MPEG7B corpus.

In [14], the authors use other two corpora for their experiments as well, a corpus of airplane shapes and a corpus of vehicle shapes. We show in Figure 28 and Table 3 the results with these corpora and our methods in comparison with the ones in [14].

8.5. A multidimensional shape descriptor

In this section, we compare the different approaches with a state-of-the-art shape descriptor different from the curvature. This descriptor uses height functions [5]. It describes each point using 20 dimensions.

Figure 29 and Table 4 show the results. As we can see our cyclic approaches obtain the best results.

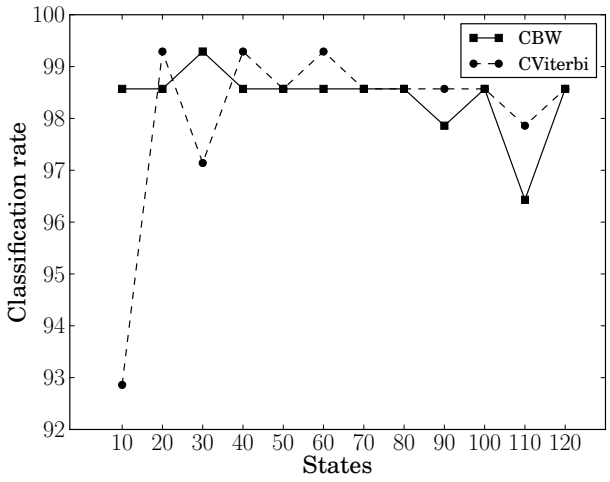
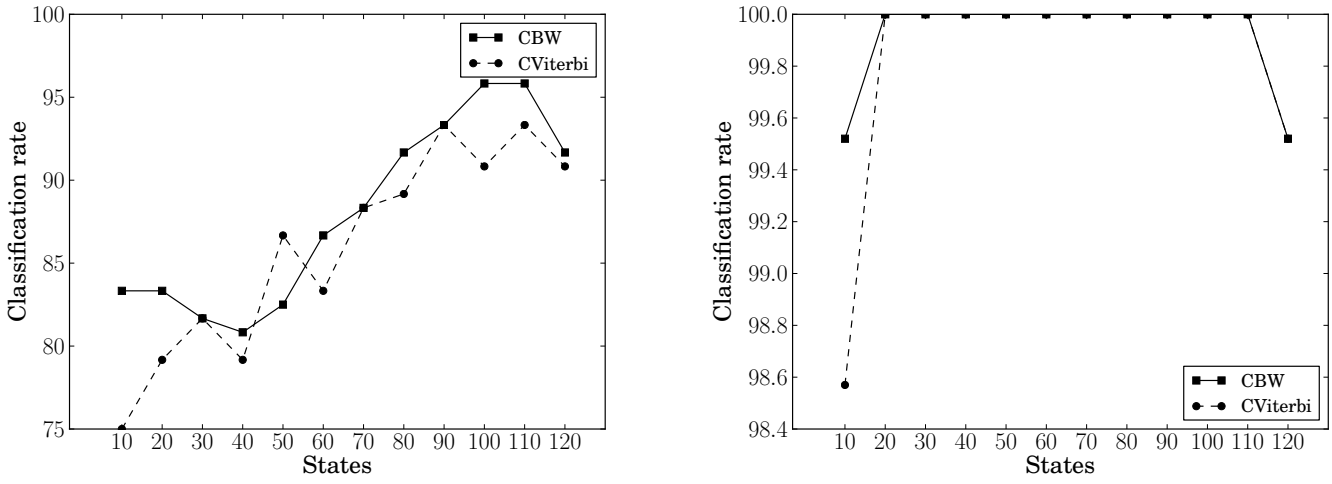


Figure 27: Classification rates with cyclic Baum-Welch (CBW) and cyclic Viterbi (CViterbi) with corpus subset 1.



(a) (b)

Figure 28: Classification rates with cyclic Baum-Welch (CBW) and cyclic Viterbi (CViterbi) with corpora: (a) vehicle shapes and (b) airplane shapes.

Table 3: Classification rates of [14] and the best results (see Figure 27 and Figure 28) for cyclic Viterbi (CViterbi) and cyclic Baum-Welch (CBW) for corpora: (a) subset 1, (b) vehicle shapes and (c) airplane shapes. Bold entries show the best results in the comparison.

Corpus	**Method**		
	GPD+Ergodic [14]	CViterbi	CBW
Subset 1	96.43	**99.29**	**99.29**
Vehicle shapes	84.17	93.33	**95.83**
Airplane shapes	99.05	**100.00**	**100.00**

Height functions together with CDTW (a distance-based method) [5] obtain a classification rate of 98.71%. In our case, using this shape descriptor together with the cyclic Viterbi approach, we achieve a 96.30%. It is worth noting that although the results using CDTW are better, we obtain competitive results considering the substantially smaller space and time costs of classification.

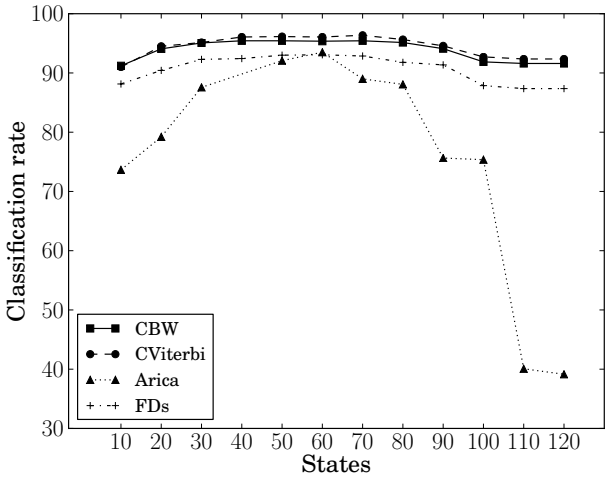


Figure 29: Classification rates (using height functions) for the comparison between: cyclic Baum-Welch (CBW), cyclic Viterbi (CViterbi), circular topology (Arica) and Fourier descriptors (FDs). With corpus MPEG7B.

9. Discussion

In this work, we have argued and empirically proved that other proposals in the literature for obtaining the invariance to the starting point do not offer a suitable solution.

Table 4: Classification rates (using height functions) of the best results (see Figure 29) for the comparison between: cyclic Baum-Welch (CBW), cyclic Viterbi (CViterbi), circular topology (Arica) and Fourier Descriptors. With corpus MPEG7B. Bold entry shows the best result in the comparison.

Corpus	**Method**			
	FDs	Arica	CViterbi	CBW
MPEG7B	93.07	93.50	**96.36**	95.43

We have experimentally proved that the linear left-to-right topology is enough for recognizing contours, and with this, it is possible to use our divide-and-conquer algorithm for this topology to speed up training and classification.

We have formalized cyclic training and cyclic recognition, formulating the cyclic Baum-Welch and the cyclic Viterbi algorithms. We have shown that this cyclic treatment is the best solution for obtaining the starting point invariance.

Considering that we use a statistical model for representing each category we obtain competitive results.

Acknowledgments

Work partially supported by the Spanish Government (TIN2010-18958) and the Generalitat Valenciana (*Prometeo*/2010/028).

[1] D. Zhang, G. Lu, Review of shape representation and description techniques, Pattern Recognition 37 (2004) 1–19.

[2] D. Sankoff, J. Kruskal (Eds.), Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison, Addison-Wesley, Reading, MA, 1983.

[3] T. Adamek, N. E. O'Connor, A multiscale representation method for nonrigid shapes with a single closed contour, IEEE Trans. Circuits Syst. Video Techn 14 (5) (2004) 742–753.

[4] H. Ling, D. W. Jacobs, Shape classification using the inner-distance, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2) (2007) 286–299.

[5] J. Wang, X. Bai, X. You, W. Liu, L. Latecki, Shape matching and classification using height functions, Pattern Recognition Letters 33 (2) (2012) 133–143.

[6] V. Palazón-González, A. Marzal, On the dynamic time warping of cyclic sequences for shape retrieval, Image and Vision Computing 30 (12) (2012) 978–990.

[7] L. R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc IEEE 77 (2).

[8] Y. He, A. Kundu, 2-D shape classification using hidden Markov model, IEEE Trans. Pattern Anal. Mach. Intell. 13 (11) (1991) 1172–1184.

[9] A. Fred, J. Marques, P. Jorge, Hidden markov models vs syntactic modeling in object recognition, in: ICIP, Vol. I, 1997, pp. 893–896.

[10] N. Arica, F. Yarman-Vural, A shape descriptor based on circular hidden Markov model, in: ICPR, Vol. I, 2000, pp. 924–927.

[11] J. Cai, Z.-Q. Liu, Hidden Markov models with spectral features for 2D shape recognition, IEEE Trans. Pattern Anal. Mach. Intell. 23 (12) (2001) 1454–1458.

[12] M. Bicego, V. Murino, Investigating hidden Markov models' capabilities in 2D shape classification, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2) (2004) 281–286.

[13] Bicego, Murino, Figueiredo, Similarity-based classification of sequences using hidden Markov models, Pattern Recognition 37 (2004) 2281–2291.

[14] N. Thakoor, J. Gao, S. Jung, Hidden Markov model-based weighted likelihood discriminant for 2-D shape classification, IEEE Trans. Image Processing 16 (11) (2007) 2707–2719.

[15] V. Palazón, A. Marzal, J. M. Vilar, Cyclic linear hidden Markov models for shape classification, in: PSIVT, 2007, pp. 152–165.

[16] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm (with discussion), Journal of the Royal Statistical Society (Series B) 39 (1) (1977) 1–38.

[17] C. Wu, On the convergence properties of the EM algorithm, Ann. Stat. 11 (1983) 95–103.

[18] G. D. Forney, The Viterbi algorithm, Proc IEEE 61 (1973) 268–278.

[19] B. K. P. Horn, Robot Vision, MIT Press, Cambridge, Massachusetts, 1986.

[20] A. Folkers, H. Samet, Content-based image retrieval using fourier descriptors on a logo database, in: ICPR (3), 2002, pp. 521–524.
URL http://doi.ieeecomputersociety.org/10.1109/ICPR.2002.1047991

[21] I. Bartolini, P. Ciaccia, M. Patella, WARP: Accurate retrieval of shapes using phase of fourier descriptors and time warping distance, IEEE Trans. Pattern Anal. Mach. Intell. 27 (1) (2005) 142–147.

[22] M. Maes, On a cyclic string-to-string correction problem, Information Processing Letters 35 (1990) 73–78.

[23] L. E. Baum, J. A. Eagon, An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model of ecology., Bull. Amer. Math. Soc. 73 (1967) 360–363.

[24] S. E. Levinson, L. R. Rabiner, M. M. Sondhi, An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition, The Bell System Technical Journal 62 (4) (1983) 1035–1074.

[25] B. H. Juang, L. R. Rabiner, The segmental K-means algorithm for estimating parameters of hidden Markov models, IEEE Trans. on Acoustics, Speech, and Signal Processing 38 (9) (1990) 1639.

[26] W. I. Zangwill, Nonlinear Programming. A Unified Approach, Prentice-Hall, Englewood Cliffs, NJ, 1969.

[27] S. Young, J. Odell, D. Ollason, V. Valtchev, P. Woodland, The HTK Book, Cambridge University 1996.

[28] L. Latecki, R. Lakämper, U. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: CVPR, IEEE, Los Alamitos, 2000, pp. 424–429.

[29] D. Sharvit, J. Chan, H. Tek, B. B. Kimia, Symmetry-based indexing of image databases, in: Workshop on Content-Based Access of Image and Video Libraries, 1998, pp. 56–62.

[30] R. O. Duda, P. E. Hart, Pattern Classification and Scene Analysis, Wiley, 1973.

[31] G. Schwarz, Estimating the dimension of a model, Ann. Stat. 14 (1978) 461–64.