On the Dynamic Time Warping of Cyclic Sequences for Shape Retrieval

Vicente Palazón-González, Andrés Marzal

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

Abstract

In the last years, in shape retrieval, methods based on Dynamic Time Warping and sequences where each point of the contour is represented by elements of several dimensions have had a significant presence. In this approach each point of the closed contour contains information with respect to the other ones, this global information is very discriminant. The current state-of-the-art shape retrieval is based on the analysis of these distances to learn better ones.

These methods are robust to noise and invariant to transformations, but, they obtain the invariance to the starting point with a brute force cyclic alignment which has a high computational time. In this work, we present the Cyclic Dynamic Time Warping. It can obtain the cyclic alignment in $O(n^2 \log n)$ time, where n is the size of both sequences. Experimental results show that our proposal is a better alternative than the brute force cyclic alignment and other heuristics for obtaining this invariance.

Keywords:
Cyclic sequences, cyclic strings, dynamic time warping, shape retrieval, shape recognition.

1. Introduction

Content-based image retrieval is being increasingly demanded in many applications: digital libraries, broadcast media selection, multimedia editing, etc. [1]. The MPEG-7 standard, for instance, is a specification for multimedia content description that defines visual feature descriptors aimed at image retrieval based on their own visual content rather than text [2]. The shape of 2D objects (written characters, trademarks, pre-segmented object contours, 2D/3D object boundaries,

Email addresses: `palazon@lsi.uji.es` (Vicente Palazón-González), `amarzal@lsi.uji.es` (Andrés Marzal)

etc.) usually provides a more powerful semantical clue for similarity matching rather than color or texture: humans can recognize characteristic objects from their shape boundary.

Shapes can be represented by their contours using shape descriptors, which can be considered as sequences. These shape descriptors can be formed by concatenation of Freeman codes, 2D points, curvature, etc. Given two of these sequences, the comparison between them is a process which measures how much they differ. Sometimes a natural alignment exists between the elements of both sequences and we can compare them by using a simple distance, such as the Euclidean distance. However, in most situations this does not happen because sequences can suffer some kind of corruption. For this reason, it is necessary to find a correspondence over all possible correspondences between the elements of the sequences. To achieve this there are efficient methods based on dynamic programming [3] such as Edit Distance (ED) [4] and Dynamic Time Warping (DTW) [5].

These (dis)similarity measures combined with shape descriptors must be robust to noise and invariant to transformations such as translation, scaling, rotation, etc. Most of these distortions are relatively easy to manage, but independently of the descriptor, there is a problem which has certain difficulty: the election of the starting point to represent the closed contour as a sequence[1]. To solve this there are two approaches: selecting a reference rotation or comparing both sequences by considering every possible starting point. The idea of the selection of a reference rotation [7, 8, 9, 10] consists of finding a *canonical* rotation for every shape and, from it, selecting a starting point with a defined criteria. However, these techniques in certain situations fail and a good alignment is crucial to achieve suitable results. We can obtain this correct alignment by measuring distances between every possible starting point, that is to say, to do a *cyclic alignment*. From this the concept of *cyclic sequence* arises. Cyclic sequences are sequences of values (or elements with several values or dimensions) which have no beginning or end. A cyclic sequence models a set with every possible cyclic shift of the sequence and measuring distances between two cyclic sequences is equivalent to measuring distances between every possible starting of both sequences.

In the last years, in shape retrieval, methods based on DTW and sequences where each point of

[1]In this paper, we only consider closed contours with no occlusions, for a more general approach we refer the reader to [6].

the contour is represented by elements of several dimensions have had a significant presence [11, 12, 13, 14, 15, 16]. In these methods each point of the closed contour contains information with respect to the other ones. This global information is very discriminant as its results have shown. The current state-of-the-art shape retrieval is based on the analysis of the distances that these methods offer to produce other distances that increase the discriminability between different shape groups, what is called context-sensitive learning [16, 17, 18, 19]. In general, all of these research accepts that in order to obtain a good starting point there must be a cyclic alignment, but they use brute force cyclic alignment which has a $O(n^3)$ computational cost (where n is the size of both sequences).

DTW has a high cost and if we have to compute it for every possible starting point, computational cost dramatically increases. Then it seems reasonable to speed up this cyclic alignment. In the case of the ED, Maes proposed in [20] an $O(n^2 \log n)$ time procedure to compute the Cyclic Edit Distance (CED), but, in spite of the similarities between the ED and DTW in their computation, as we see in this paper, the use of this algorithm directly with DTW is not possible.

Keogh et al. [21] also proposed a solution to speed up the cyclic alignment using an indexing approach, but it seems to be suitable just for sequences of one dimension and not, for instance, 96 dimensions as [14] needs.

In this paper, we present an algorithm inspired by [20], the Cyclic Dynamic Time Warping (CDTW). It can obtain the cyclic alignment of DTW in $O(n^2 \log n)$ time and can deal with any number of dimensions in the elements of the sequences. This algorithm significantly speeds up shape retrieval tasks.

This paper is organized as follows. In Section 2, the Edit Distance and the Dynamic Time Warping are revisited. In Section 3, the drawbacks of canonical shift methods are pointed out. A CDTW procedure that provides starting point invariance when comparing sequences of shape descriptors is presented in Section 4. In Section 5, final considerations to take into account in shape retrieval with the proposed method are mentioned. In Section 6, experimental results on image retrieval tasks for several databases compare the different methods. Finally, some conclusions are presented in Section 7.

2. Sequence Comparison: Edit Distance and Dynamic Time Warping

Let $A = a_0a_1 \dots a_{m-1}$ and $B = b_0b_1 \dots b_{n-1}$ be two sequences in Σ^*, where Σ^* is the closure under concatenation of a set Σ, and let λ denote the empty sequence, that is to say, a sequence of length 0.

An *edit operation* is a pair of sequences with a length less than or equal to 1, $(u, v) \neq (\lambda, \lambda)$, denoted by $u \to v$. Edit operations are classified as *deletions* $(u \to \lambda)$, *insertions* $(\lambda \to v)$, and *substitutions* $(u \to v)$, where u, $v \in \Sigma$. A sequence B results from another sequence A via the edit operation $u \to v$ if there are two sequences C and D such that $A = CuD$ and $B = CvD$. An *edit sequence* is a sequence of edit operations, $e = e_1e_2 \dots e_k$, that transforms A into B if B can be obtained from A by successive application of the edit operations. Edit operations can have a cost by means of a function $\gamma : (\Sigma \cup \{\lambda\}) \times (\Sigma \cup \{\lambda\}) \to \mathbb{R}^{\geq 0}$ satisfying $\gamma(u \to v) + \gamma(v \to z) \geq \gamma(u \to z)$. The *cost of an edit sequence* $e = e_1e_2 \dots e_k$ is defined as $\gamma(e) = \sum_{1 \leq i \leq k} \gamma(e_i)$. An optimal edit sequence from A to B is an edit sequence of minimum cost that transforms A into B. The Edit Distance (ED) between A and B is denoted with $ED(A, B)$ and is defined as the cost of an optimal edit sequence from A to B.

Wagner and Fisher [4] showed that $ED(A, B) = d(m, n)$, where

$$d(i,j) = \begin{cases} 0, & \text{if } i = j = 0; \\ d(i-1, 0) + \gamma(a_{i-1} \to \lambda), & \text{if } i > 0 \text{ and } j = 0; \\ d(0, j-1) + \gamma(\lambda \to b_{j-1}), & \text{if } i = 0 \text{ and } j > 0; \\ \min \left\{ \begin{array}{l} d(i-1, j) + \gamma(a_{i-1} \to \lambda), \\ d(i, j-1) + \gamma(\lambda \to b_{j-1}), \\ d(i-1, j-1) + \gamma(a_{i-1} \to b_{j-1}), \end{array} \right\} & \text{otherwise.} \end{cases} \quad (1)$$

They also proposed a Dynamic Programming procedure to compute $ED(A, B)$ in $O(mn)$ time [4] by computing the cost of an optimal path in a directed, acyclic graph: the so-called *edit graph.* This graph is a grid of nodes (i, j), where $0 \leq i \leq m$ and $0 \leq j \leq n$, connected by horizontal, vertical and diagonal arcs, as can be seen in Figure 1. The horizontal arc departing from node (i, j) represents $a_i \to \lambda$, the vertical arc represents $\lambda \to b_j$, and the diagonal arc represents $a_i \to b_j$. Each path from $(0, 0)$ to (m, n) is an *edit path* and its cost is the cost of its associated

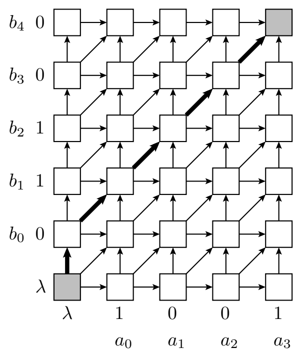

Figure 1: Edit graph for $A = 1001$ and $B = 01100$, where $\gamma(0 \to 1) = 1$ and $\gamma(0 \to 0) = \gamma(1 \to 1) = 0$. The optimal edit path is shown with thick arrows.

edit sequence.

Edit operations do not naturally arise in all sequence comparison problems. Optimal alignment (also known as Dynamic Time Warping) leads to a better dissimilarity measure when we want to model "elastic deformations". An *alignment* between A and B is a sequence of pairs (i_0, j_0), (i_1, j_1), $\ldots$, (i_{k-1}, j_{k-1}) such that (a) $0 \le i_\ell < m$ and $0 \le j_\ell < n$ for $0 \le \ell < k$; (b) $0 \le i_{\ell+1} - i_\ell \le 1$ and $0 \le j_{\ell+1} - j_\ell \le 1$ for $0 \le \ell < k-1$; and (c) $(i_\ell, j_\ell) \ne (i_{\ell+1}, j_{\ell+1})$ for $0 \le \ell < k-1$. The pair (i_ℓ, j_ℓ) is said to *align* a_{i_ℓ} with b_{j_ℓ}. The *cost of an alignment* is defined as $\sum_{0 \le \ell < k} \delta(a_{i_\ell}, b_{j_\ell})$, where δ is the local distance function. An optimal alignment is an alignment of minimum cost. Figure 2a shows an optimal alignment between two sequences.

The Dynamic Time Warping (DTW) dissimilarity measure, $DTW(A, B)$, is defined as the cost of the optimal alignment between A and B, that is to say, $DTW(A, B) = d'(m-1, n-1)$[5], where

$$d'(i,j) = \begin{cases} \delta(a_0, b_0), & \text{if } i = j = 0; \\ d'(i-1, 0) + \delta(a_i, b_0), & \text{if } i > 0 \text{ and } j = 0; \\ d'(0, j-1) + \delta(a_0, b_j), & \text{if } i = 0 \text{ and } j > 0; \\ \min \left\{ \begin{array}{l} d'(i-1, j-1), \\ d'(i-1, j), \\ d'(i, j-1) \end{array} \right\} + \delta(a_i, b_j), & \text{otherwise.} \end{cases} \quad (2)$$

This equation can be solved by Dynamic Programming in $O(mn)$ time: the problem is reduced to the computation of an optimal path in the *warping graph*, a weighted, acyclic graph with $O(mn)$ arcs. The warping graph is a grid of nodes (i, j), where $0 \le i < m$ and $0 \le j < n$, connected by horizontal, vertical and diagonal arcs as shown in Figure 2b. All arcs ending at node (i, j) have the same cost, $\delta(a_i, b_j)$. Warping paths start at node $(0, 0)$ and end at node $(m-1, n-1)$.

Alignments of pairs of symbols in DTW can be assimilated to substitutions in edit distances, but DTW allows for one-to-many correspondences, which makes it appropriate to model elastic distortions. On the other hand, DTW alignments have no insertions or deletions and may seem preferable to the Edit Distance when these operations do not naturally arise.

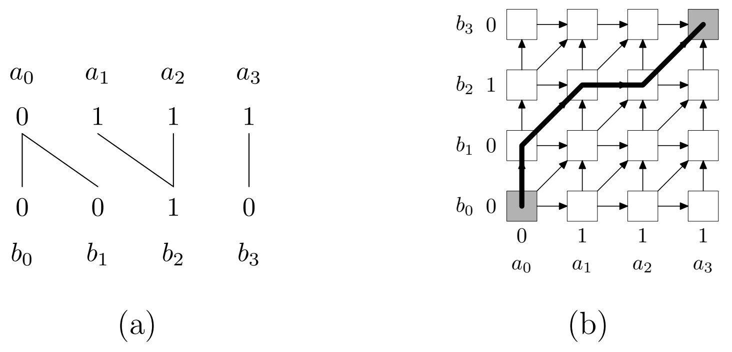

Figure 2: (a) An optimal alignment between the sequences $A = 0111$ and $B = 0010$, where $\delta(a_i, b_j) = |a_i - b_j|$. (b) Warping graph underlying the solution procedure of the recursive equation (2). The thick line is the optimal alignment associated to (a).

It is common to impose global restrictions to DTW in order to avoid an alignment too far away from the diagonal of the graph. To achieve this we can use an alignment *band* that limits which nodes can be visited. In Figure 3, there is an example of Sakoe band [22], the most widely used band. Thus, we can add a fourth constraint (being $m = n$): (d) $|i_l - j_l| \le s$ for $0 \le \ell < k$, where s is the size of the band. In Figure 3, $s = 2$.

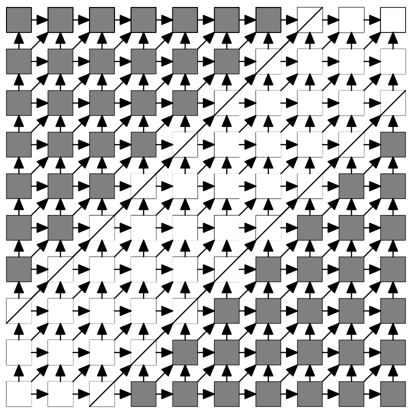

Figure 3: Sakoe band. White nodes can be visited.

There are two main reasons to use this heuristic. One of them is to speed up computation. The execution of DTW_s (DTW with a Sakoe band of size s) has a $O(\max(m,n)s)$ computational cost. Second, to prevent *pathological paths*, that is to say, paths that are aligned with a small part of one sequence to a big part of another. In certain fields of application that is not suitable.

There are also alternative definitions of the DTW with *weights* that affect the different arcs [23, 24]. We can use a function, ω, that returns the weight for each type of arc, $\omega = (w(1,0), w(1,1), w(0,1))$. The recursive equation for $DTW_\omega(A,B) = d(m-1, n-1)$ is then:

$$d(i,j) = \begin{cases} w(1,1)\cdot\delta(a_0,b_0), & \text{if } i=j=0;\\ d(i-1,0)+w(1,0)\cdot\delta(a_i,b_0), & \text{if } i>0 \text{ and } j=0;\\ d(0,j-1)+w(0,1)\cdot\delta(a_0,b_j), & \text{if } i=0 \text{ and } j>0;\\ \min\left\{\begin{array}{l} d(i-1,j-1)+w(1,1)\cdot\delta(a_i,b_j),\\ d(i-1,j)+w(1,0)\cdot\delta(a_i,b_j),\\ d(i,j-1)+w(0,1)\cdot\delta(a_i,b_j)\end{array}\right\}, & \text{otherwise.}\end{cases} \tag{3}$$

If $\omega = (1,1,1)$, $DTW_\omega(A,B) = DTW(A,B)$. These weights are particularly interesting for giving more importance to some arcs and can be useful in normalization.

Sometimes normalization by the duration of sequences is necessary because of a high difference between the size of the sequences. A possible definition of the normalized DTW (NDTW) is:

$$NDTW(A,B) = \min_{(i_0,j_0)\ldots(i_{k-1},j_{k-1})} \frac{\sum_{\ell=0}^{k-1} w(i_\ell - i_{\ell-1}, j_\ell - j_{\ell-1})\cdot\delta(a_{i_\ell}, b_{j_\ell})}{\sum_{\ell=0}^{k-1} w(i_\ell - i_{\ell-1}, j_\ell - j_{\ell-1})}, \tag{4}$$

where $w(i_o - i_{-1}, i_0 - i_{-1}) = w(1,1)$.

If we want to calculate (4) with the computational cost of DTW, denominator cannot depend on the alignment [25]. This happens, for example, with $\omega = (1,2,1)$, because in this case denominator is $m+n$ for any alignment. If denominator depends on the alignment we have to use [26].

Dynamic Time Warping does not provide invariance to changes in position, scale, orientation of contours and selection of starting points in the compared shapes. Unless a proper description of the shape is used, the DTW does not lead to good dissimilarity measures. Shape descriptors such as [12, 13, 14] provide all these invariants except the selection of the starting point.

3. Canonical Methods

Before obtaining the shape descriptor from the sequence of points, $A = a_0 a_1 \ldots a_{m-1}$, we can apply a heuristic method to select a starting point. The basic idea of this approach is to find a *canonical* rotation, and in base of this rotation to choose a starting point. This approach is suitable in restricted domains where major axis is well defined. Shape orientation can be determined by its axis of least second moment of inertia [7, 8, 27], using the following area based equation: $\tan(2\theta) = 2\mu_{11}/(\mu_{20} - \mu_{02})$, being $\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y)$, where $I(x, y)$ is a binarized image obtained with the area of the shape defined by the contour A, and $\bar{x}$ and $\bar{y}$ are the centroids. Another possibility is to use Fourier Descriptors [8, 28, 9] (FDs). The Discrete Fourier Transform of a sequence of points, A, is an ordered set of complex values $\Delta = (\alpha_{-m/2}, \ldots, \alpha_{-1}, \alpha_0, \alpha_1, \ldots, \alpha_{m/2-1})$, where $\alpha_i = \sum_{0 \leq k < m} a_k e^{-j2\pi ki/m}$ and $j = \sqrt{-1}$. These coefficients are the FDs and model the contour of a shape as a composition of ellipses revolving at different frequencies [28]. The main ellipse is centered at the contour centroid, α_0, and translation of the contour only affects this descriptor. Scaling the FDs by a factor scales the shape by the same factor. Rotating the shape by an angle θ yields a phase shift of θ in the FDs. Changing the starting point k symbols, that is to say, working on the cyclic shift $a_k a_{k+1} \ldots a_{m-1} a_0 a_1 \ldots a_{k-1}$, produces a linear phase shift of $2\pi ki/m$ to α_i, for $-m/2 \leq i < m/2$. Invariance to rotation can be obtained by substracting $(\theta_{-1} + \theta_1)/2$ (the orientation of the basic ellipse) to each θ_i. Invariance with respect to the starting point can be achieved by adding $i(\theta_{-1} - \theta_1)/2$ to each θ_i. The shape can be reconstructed to a canonical form.

It should be noted that subtracting $(\theta_{-1} + \theta_1)/2$ to the orientation of all FDs only provides rotation invariance modulo π radians [28], there is an ambiguity. Anyway, let us consider that the rotation ambiguity is not present. The basic idea of the FDs method is that, after normalization, all shapes have a canonical version with a "standard" rotation and starting point, and thus, they can be compared by means of the DTW dissimilarity measure. 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

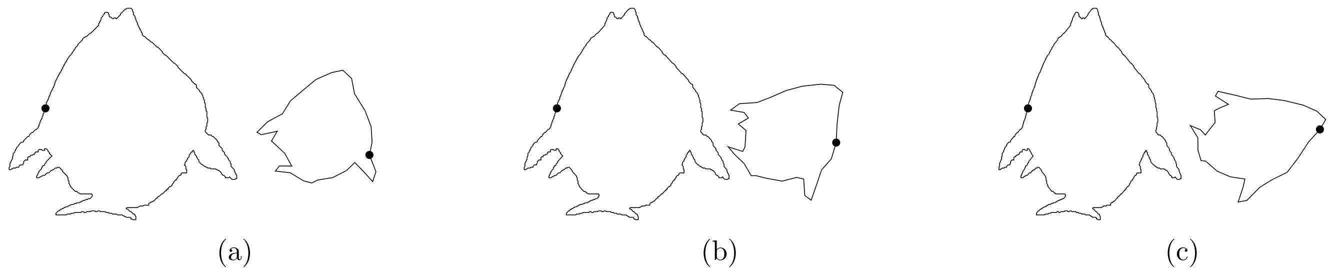
Figure 4: (a) Original shape and its canonical version using FDs. (b) The same shape compressed in the horizontal axis and its canonical version, which has a different rotation and starting point. (c) A slightly more compressed shape and its canonical version, which is also different.

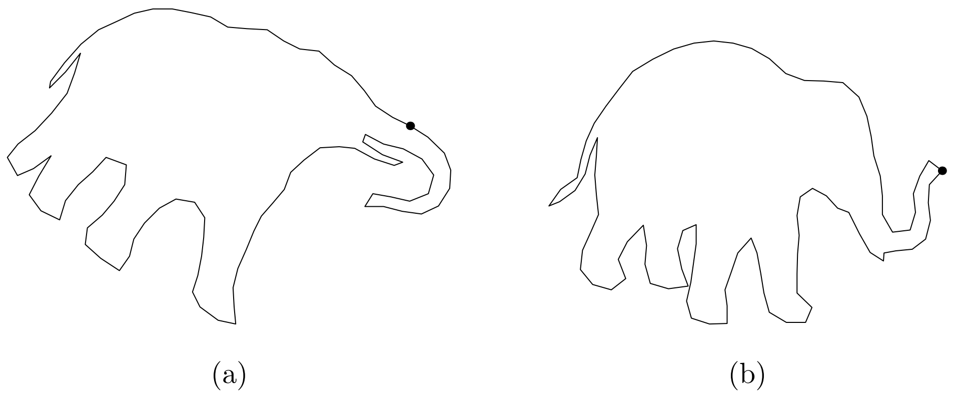
Figure 5: (a) Canonical version of an elephant with its trunk down. (a) Canonical version of an elephant with its trunk raised. Both canonical versions have been obtained by the method of least second moment of inertia.

basic ellipse is almost a circle. Besides, shapes of the same category with little differences can substantially alter the starting point selection. 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 then, the selection of the starting point.

Although in the bibliography there are other methods, all of them have these problems. Therefore, invariance to starting point election should be provided by a different method.

4. Cyclic Sequence Comparison: Cyclic Edit Distance and Cyclic Dynamic Time Warping

When two signatures have "equivalent" starting points, DTW provides a good dissimilarity measure. However, considering that similar shapes can present very different starting points, it is useful to consider the problem under the framework of *cyclic sequences*.

A cyclic shift σ of a sequence $A = a_0 a_1 \ldots a_{m-1}$ is a mapping $\sigma : \Sigma^* \rightarrow \Sigma^*$ defined as $\sigma(a_0 a_1 \ldots a_{m-1}) = a_1 \ldots a_{m-1} a_0$. Let σ^k denote the composition of k cyclic shifts and let σ^0 denote the identity. Two sequences A and A' are cyclically equivalent if $A = \sigma^k(A')$, for some k. The equivalence class of A is $[A] = \{\sigma^k(A) : 0 \leq k < m\}$, denoted as cyclic sequence, and any of its members is a representative (non-cyclic) sequence. For instance, let 1 and 0 be two elements from the set Σ; the set $\{0111, 1110, 1101, 1011\}$ is a cyclic sequence and 1101 —or any other sequence in the set— can be taken as its representative.

The *Cyclic Edit Distance* (CED) between $[A]$ and $[B]$ is defined as

$$CED([A],[B]) = \min_{0 \leq k < m} \left(\min_{0 \leq \ell < n} ED(\sigma^k(A), \sigma^\ell(B)) \right). \tag{5}$$

Maes [20] showed the following lemma:

Lemma 1 (Maes). $CED([A],[B]) = CED([A],B) = \min_{0 \leq k < m} ED(\sigma^k(A), B)$. $\square$

Since $ED(\sigma^k(A), B)$ can be computed in $O(mn)$ time, the value of $CED([A],[B])$ can be obtained by computing m edit distances in $O(m^2 n)$ time. Maes proposed a more efficient procedure that computes m shortest paths in an *extended edit graph*, an edit graph where one of the sequences is doubled (see Figure 6a). Let $P(k)$ be a shortest path between nodes $(k, 0)$ and $(k+m, n)$ in the extended edit graph. The edit distance $ED(\sigma^k(A), B)$ is the cost of $P(k)$. When computing $ED(\sigma^k(A), B)$, one can take advantage of the non-crossing property of edit paths [20] (see Figure 6b):

Property 1 (non-crossing property, Maes). *Let j, k, and l be three integers such that $0 \leq j < k < l \leq m$, and let $P(j)$ and $P(l)$ be two non-crossing paths with minimum cost in the extended edit graph. There is a shortest path $P(k)$ from $(k, 0)$ to $(k+m, n)$ that lies between $P(j)$ and $P(l)$.*

This property leads to a Divide and Conquer [3], recursive procedure: when $P(j)$ and $P(l)$ are known, $P((j+l)/2)$ is computed by only taking into account those nodes of the extended edit graph lying between $P(j)$ and $P(l)$; then, optimal paths bounded by $P(j)$ and $P((j+l)/2)$ and optimal paths bounded by $P((j+l)/2)$ and $P(l)$ can be recursively computed. The recursive procedure starts after computing $P(0)$ (by means of the standard Edit Distance) and $P(m)$, which

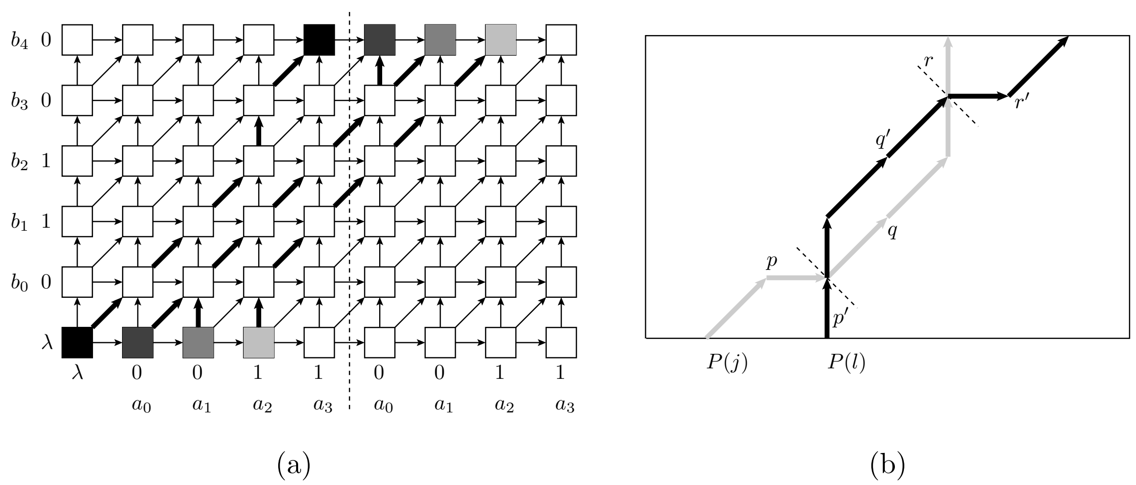

Figure 6: (a) Extended edit graph for $A = 0011$ and $B = 01100$. The optimal path for the cyclic edit distance of A and B is the optimal path among those ones starting and ending at nodes with the same colour. (b) $P(j)$ is the optimal edit path for $\sigma^j(A)$ and B, and $P(l)$ is the optimal path for $\sigma^l(A)$ and B. Crossing paths can be avoided: if the cost of q is greater than the cost of q', $P(j)$ can be improved by taking q' instead of q.

is $P(0)$ 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. Total computation time is, therefore, $O(mn \log m)$.

A *cyclic alignment* between $[A]$ and $[B]$ is a sequence of pairs (i_0, j_0), (i_1, j_1), $\ldots$, (i_{k-1}, j_{k-1}) such that, for $0 \leq \ell < k$, (a) $0 \leq i_\ell < m$ and $0 \leq j_\ell < n$; (b) $0 \leq (i_{(\ell+1) \bmod k} - i_\ell) \bmod m \leq 1$ and $0 \leq (j_{(\ell+1) \bmod k} - j_\ell) \bmod n \leq 1$; and (c) $(i_\ell, j_\ell) \neq (i_{(\ell+1) \bmod k}, j_{(\ell+1) \bmod k})$. The *cost of a cyclic alignment* (i_0, j_0), (i_1, j_1), $\ldots$, (i_{k-1}, j_{k-1}) is defined as $\sum_{0 \leq \ell < k} \delta(a_{i_\ell}, b_{j_\ell})$, where δ is the local dissimilarity function. An optimal cyclic alignment is a cyclic alignment of minimum cost.

The Cyclic Dynamic Time Warping (CDTW) dissimilarity measure $CDTW([A], [B])$ is defined as the cost of the optimal cyclic alignment between A and B. First, we show that the optimal cyclic alignment can be defined in terms of alignments between non-cyclic sequences, i.e., in terms of $DTW(\cdot, \cdot)$; then, we present how to adapt the Maes' algorithm to compute it.

Lemma 2. *If $m, n > 1$ and (i_0, j_0), (i_1, j_1), $\ldots$, (i_{k-1}, j_{k-1}) is an optimal alignment between two sequences $a_0 a_1 \ldots a_{m-1}$ and $b_0 b_1 \ldots b_{n-1}$, there is at least one ℓ such that $i_\ell \neq i_{(\ell+1) \bmod k}$ and $j_\ell \neq j_{(\ell+1) \bmod k}$.*

Proof: Any alignment including (i_ℓ, j_ℓ), $(i_\ell{+}1, j_\ell)$, and $(i_\ell{+}1, j_\ell{+}1)$ can be "improved" by removing

$(i_\ell + 1, j_\ell)$, since $\delta(a_{i_\ell+1}, b_{j_\ell}) \geq 0$. Analogously, any alignment including (i_ℓ, j_ℓ), $(i_\ell, j_\ell + 1)$, and $(i_\ell + 1, j_\ell + 1)$ can be "improved" by removing $(i_\ell, j_\ell + 1)$. $\square$

Lemma 3. *The CDTW dissimilarity between a cyclic sequence $[A]$ of length m and a cyclic sequence $[B]$ of length n, $CDTW([A], [B])$, can be computed as:*

$$CDTW([A], [B]) = \min_{0 \leq k < m} \left(\min_{0 \leq \ell < n} DTW(\sigma^k(A), \sigma^\ell(B)) \right). \tag{6}$$

Proof: Trivial when $m = 1$ or $n = 1$. Let us consider that $m, n > 1$ and let (i_0, j_0), (i_1, j_1), $\ldots$, (i_{k-1}, j_{k-1}) be an optimal alignment. Let ℓ be an index such that $i_\ell \neq i_{(\ell+1) \bmod k}$ and $j_\ell \neq j_{(\ell+1) \bmod k}$ (by Lemma 2). The cost of this cyclic alignment is $DTW(\sigma^{i_{(\ell+1) \bmod k}}(A), \sigma^{j_{(\ell+1) \bmod k}}(B))$, which is considered by the double minimization, since $0 \leq i_{(\ell+1) \bmod k} < m$ and $0 \leq j_{(\ell+1) \bmod k} < n$. $\square$

According to Lemma 3, the value of $CDTW([A], [B])$ can be trivially computed in $O(m^2 n^2)$ time by performing all cyclic shifts on both sequences, solving mn recurrences like equation (2). Now the question is: Is it possible to perform cyclic shifts on only one of the sequences when computing the CDTW like Maes' algorithm does for the CED? The answer is negative: in general, $CDTW([A], [B])$ is neither $\min_{0 \leq k < m} DTW(\sigma^k(A), B)$ nor $\min_{0 \leq l < n} DTW(A, \sigma^l(B))$, as the following counter-example shows: being $\delta(a_i, b_j) = |a_i - b_j|$; the value of $CDTW([101], [010])$ is 0, since $DTW(110, 100) = 0$, but $DTW(101, 010) = 2$ and $DTW(011, 010) = DTW(110, 010) = DTW(101, 100) = DTW(101, 001) = 1$. Therefore, an equivalent of Maes' algorithm for the CED computation cannot be directly applied to CDTW dissimilarity computation.

However, a significant speed-up can be achieved as a consequence of the next theorem:

Theorem 1. *The CDTW dissimilarity between cyclic sequences $[A]$ and $[B]$ can be computed as:*

$$CDTW([A], [B]) = \min_{0 \leq k < m} \left(\min(DTW(\sigma^k(A), B), DTW(\sigma^k(A) a_k, B)) \right). \tag{7}$$

Proof: Each alignment induces a segmentation on A and a segmentation on B. All the values in a segment are aligned with the same value of the other cyclic sequence (Lemma 2). There is a problem when $b_{n-p-1} b_{n-p} \ldots b_{n-1}$ and $b_0 b_1 \ldots b_q$, for some p, $q \geq 0$, should belong to the same segment of b. In that case, the optimal path cannot be obtained by simply shifting a, since b_{n-1}

must be aligned with the last value of $\sigma^k(A)$ and b_0 must be aligned with its first value, i.e., they cannot belong to the same segment. The sequence $\sigma^k(A)a_k$ allows the alignment of $b_{n-p}b_{n-p+1}\dots b_n$ and $b_0b_1\dots b_q$ with the first value of $\sigma^k(A)$, since a_k also appears at the end of $\sigma^k(A)a_k$. $\square$

Corollary 1. *$DTW(\sigma^k(A), B)$, for each k, can be obtained as a subproduct of the computation of $DTW(\sigma^k(A)a_k, B)$.*

Proof: The warping graph underlying $DTW(\sigma^k(A), B)$ is a subgraph of the graph underlying $DTW(\sigma^k(A)a_k, B)$. The optimal path in $DTW(\sigma^k(A), B)$ is a path in $DTW(\sigma^k(A)a_k, B)$. $\square$

In Figure 7 we can see this fact. Besides, it depicts that now the counter-example is not.

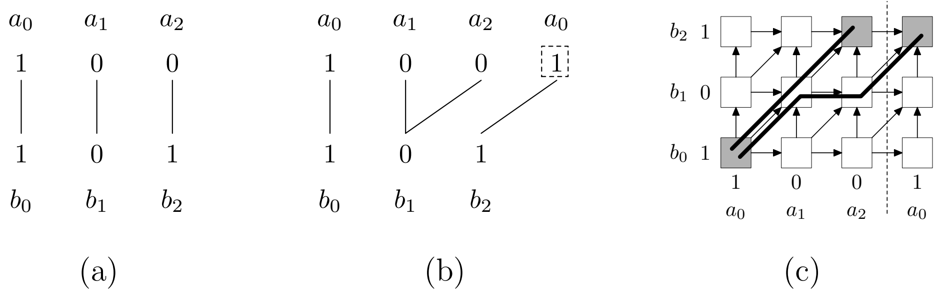

Figure 7: Counter-example mentioned before where the minimum cost is obtained using the proposed solution. (a) Optimal alignment between the sequences $\sigma^k(A) = 100$ and $B = 101$, where $\delta(a_i, b_j) = |a_i - b_j|$. (b) Optimal alignment between the sequences $\sigma^k(A)a_k = 1001$ and $B = 101$. (c) The optimal paths of $DTW(\sigma^k(A), B)$ and $DTW(\sigma^k(A)a_k, B)$.

The value of $DTW(\sigma^k(A), B)$ and $DTW(\sigma^k(A)a_k, B)$, for each k, can be obtained by computing shortest paths in an *extended warping graph* similar to the extended edit graph defined by Maes [20] (see Figure 8). Since the non-crossing property of edit paths also holds for alignment paths (see Figure 6b), the divide-and-conquer approach proposed by Maes can be applied to CDTW.

Theorem 2. *$CDTW([A], [B])$ can be computed in time $O(mn \log m)$ using the algorithm in Figure 9.*

Proof: It should be taken into account that, unlike in Maes' algorithm, the optimal path starting at $(k, 0)$ can finish either at node $(k + m - 1, n - 1)$ or $(k + m, n - 1)$, in other words, we have two paths for each starting node (see Figure 8). Which one should we take for the following recurrences? In fact, it is indifferent. Due to the non-crossing property, the two new paths that we have to compute in each recurrence cannot cross any of these paths. The divide-and-conquer procedure only needs one path as a limit on both the left and the right, then for each starting node we only need one of them and it does not matter which one. However, to follow a unified criterion we take the path with the minimum cost.

The running time of this algorithm is $O(mn \log m)$: each recursive step divides the search space in two halves and all recursive operations at the same recursion level require total $O(mn)$ time (see Figure 10). □

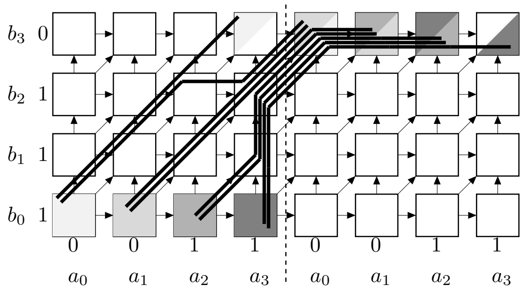

Figure 8: Extended warping graph for $A = 0011$ and $B = 1110$, where $\delta(a_i, b_j) = |a_i - b_j|$. Arcs ending at node (i, j) have a cost $\delta(a_i, b_j)$. The optimal alignment for $[A]$ and $[B]$ is the path with minimum cost starting from any colored node in the lower row and ending at a node containing the same color in the upper row (all candidate paths are shown with thick lines).

The divide-and-conquer procedure greatly reduces computation time. But, in the CED there is another procedure that reduces even more this time [29]. In this approach, lower bounds based on the cyclic shift of the sequences are used. The difference of the ED between two sequences and the ED between one of these sequences and a cyclic shift of the other is bounded by an insertion and a deletion:

$$|ED(\sigma^{k+1}(A), B) - ED(\sigma^k(A), B)| \leq \gamma(\lambda \to v) + \gamma(u \to \lambda). \tag{8}$$

As [29] explains, this Branch-and-Bound [3] procedure dismisses the calculation between two optimal paths when the lower bound is greater than the minimum cost calculated so far, which for the CED results in an extremely fast method. Nevertheless, this procedure cannot be applied to CDTW. When we make a cyclic shift in a sequence the difference can not be bounded this way, because a shift can completely disrupt the alignment and then the cost of it. It may be seen more

Figure 9: CDTW Algorithm.

```
Input: $A, B$: sequences
Output: $d^* : \mathbb{R}$
var $P$: vector $[0..m]$ alignment paths
begin
    $d^* = \min(DTW(\sigma^0(A), B), DTW(\sigma^0(A)A_k, B))$
    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
        $d^* =$ min($d^*$, NextStep($A \cdot A$, $B$, $0$, $m$))
    return $d^*$
end

function NextStep($AA$: sequence, $B$: sequence, $l : \mathbb{N}$, $r : \mathbb{N}$):$\mathbb{R}$
begin
    $c = l + \lceil \frac{r-l}{2} \rceil$
    $d = \min(DTW(AA_{c:c+m}, B), DTW(AA_{c:c+m+1}, B))$ with $P[l]$ and $P[r]$ known
    if $l + 1 < c$ then
        $d =$ min($d$, NextStep($AA$, $B$, $l$, $c$))
    if $c + 1 < r$ then
        $d =$ min($d$, NextStep($AA$, $B$, $c$, $r$))
    return $d$
end
```

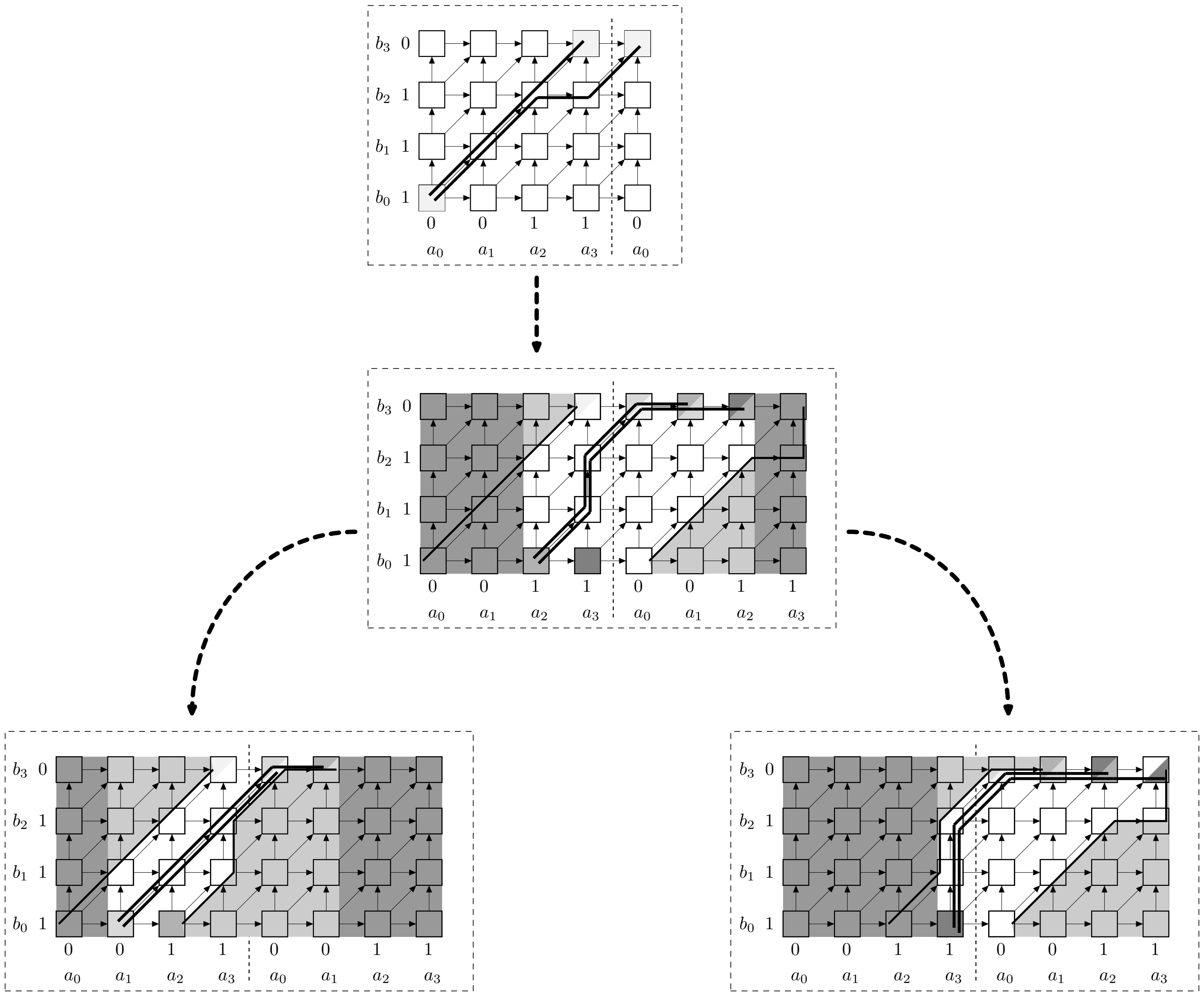

Figure 10: Divide-and-conquer procedure to compute the CDTW dissimilarity between the sequences of Figure 8. First, the optimal alignment (path) between A and B and between $\sigma^0(A)a_0$ and B is computed. The first optimal path is used as a left and right frontier in the extended graph: only the white region must be explored to compute the optimal alignment between $\sigma^2(A)$ and B and between $\sigma^2(A)a_2$ and B. This idea is recursively applied to the computation of the other optimal alignments, but using also the optimal alignment between $\sigma^2(A)$ and B as a new left or right frontier.

clearly explained by the following example. Let $A = 1110$ e $B = 0001$ be two sequences, where $DTW(A, B) = DTW(1110, 0001) = 3$, being $\delta(a_i, b_i) = |a_i - b_i|$. If we make a cyclic shift in A the following happens, $DTW(\sigma^1(A), B) = DTW(0111, 0001) = 0$. In DTW there are no deletions or insertions, only alignments and to bound them with an operation, at least linear, does not seem possible.

As we mention in Section 2, using global restrictions in DTW can speed up computation and/or prevent pathological paths.

Theorem 3. *CDTW with Sakoe band between a two cyclic sequences, $[A]$ and $[B]$ can be computed with:*

$$CDTW_s([A], [B]) = \min_{0 \le k < m} \Big(\min(DTW_s(\sigma^k(A), B), DTW_s(\sigma^k(A)a_k, B)) \Big). \tag{9}$$

Proof: Trivial from the definition of the Sakoe band in Section 2 and Theorem 1. □

Theorem 4. *The divide-and-conquer procedure can be used to obtain CDTW with Sakoe band in time $O(sn \log m)$.*

Proof: For each cyclic shift (or each recurrence of the algorithm) we use a band to the left that we have to combine with the limit on the left. The same for the band and the limit on the right. Combinations lead to new limits on the left and the right. Obviously, for the limit on the left we take the position nearest to the right (path limit or band), the opposite for the limit on the right. With these new determined limits we obtain the optimal path starting at $(k, 0)$ and finishing at node $(k+m-1, n-1)$ or $(k+m, n-1)$. Computational time of this algorithm is finally $O(sn \log m)$. □

Although the computational time is lower than the one of CDTW without the band, we are not taking into account that there is an additional computation when we combine bands and paths in each recurrence. Consequently, computational time is only reduced for small values of s. Thus, Sakoe band is only useful, in this case, to prevent pathological paths.

Extensions for other weights in the arcs and normalization are shown in Appendix A and Appendix B.

5. Final Considerations to Speed Up Shape Retrieval

In order to speed up cyclic DTW comparisons between the shape descriptors that are mentioned in [11, 12, 13, 14, 15, 16], we have to take into account the following.

As we mention before, these authors accept that the cyclic alignment is the best solution but they use brute force to obtain it with an $O(n^3)$ computational time, being n the size of the sequences. Sequences have the same size due to an equidistant sampling, of usually 100 points. With this sampling they avoid normalization and an intractable problem due to the high size of contours.

In [14], the authors try to avoid brute force by means of another heuristic. They say that (with their shape descriptor) it is often sufficient to try aligning a fixed number of k points, k is obtained experimentally. Therefore, the time complexity is $O(kn^2) = O(n^2)$. In their experiments, for a sampling of 100 points, $k = 8$, and for a sampling of 128 points, $k = 16$ (for different corpuses). As we can see, there is a correspondence between n and k. Our proposal is superior, we achieve a $O(n^2 \log n)$, a better time complexity and always with the correct cyclic alignment.

In [21], the authors, using a method similar to their previous work with DTW [30], try to speed up the cyclic alignment as well. They make clusters of sequences based on their similarity, treating every possible starting point as a different sequence and using indexing methods with lower bounds of these clusters. This solution seems to be suitable just for sequences with only one dimension (such as the curvature) and not for much more dimensions [11, 12, 13, 14, 15, 16]. For instance, in [14], 96 dimensions are required. Another problem is that it cannot use more sophisticated local distances (in DTW between elements of the sequences) such as χ^2 [14], due to their lower bound. Our proposal can deal with any number of dimensions and these local distances.

Another important thing is the cost of the local distance. For these shape descriptors [11, 12, 13, 14, 15, 16], that have a high dimensionality in their elements, the local distance has a significant cost in the computational time. It is not the same to compute a local distance between elements of one dimension (such as the curvature) and to compute it between elements of 60 dimensions (such as the Shape contexts [11]). For this reason, it is convenient to compute the $n \times n$ matrix of local distances at the beginning and store it, in order to avoid repeating calculations. We can do this both in brute force cyclic alignment and in CDTW. In the case of DTW with Sakoe band, the fact that not visiting some nodes makes to not having to calculate the corresponding local distances, that are out of the band (grey nodes in Figure 3). This is not possible in CDTW, there is no way to avoid the computation of local distances with the Sakoe band. For instance, in Figure 8, for a band with $s = 1$, when we compute the alignment starting at $(0, 0)$, we could think that we can avoid calculating, $\delta(a_0, b_2)$, $\delta(a_1, b_3)$ and $\delta(a_1, b_3)$, in the left side of the band. But this is not possible because we must compute those local distances in the other part of the extended graph for other alignments.

Finally, to mention that in the context-sensitive learning methods [16, 17, 18, 19], in a shape retrieval task, for a given query shape, in a first stage we need to compute the distances over the entire database and then, based on these distances, we can obtain with these methods the improved distances. Our algorithm can be used to speed up the computation of the distances in this first stage.

6. Experiments

In order to test the presented algorithm, we performed comparative experiments on a shape retrieval task using three publicly available databases:

- MPEG-7 CE-Shape-1 database part B (MPEG-7B). It contains 1400 shapes (see Figure 11) divided in 70 categories, each category with 20 images [31].
- Silhouette database [32]. It contains 1070 silhouettes (see Figure 12). The shapes belong to 41 categories representing different objects.
- SQUID Demo database [33]. It consist of 1100 contours of marine species (see Figure 13). The original database does not divide the contours into classes. We used the labels of [9], they manually classified 252 images into 10 semantic categories.

All the outer closed contours of the images were extracted as sequences of points. A random starting point in the sequences were also selected and 100 landmark points were sampled uniformly. We used three well-known shape descriptors, given the sequence of landmark points:

- Curvature [34]. A descriptor with only one dimension for each point.

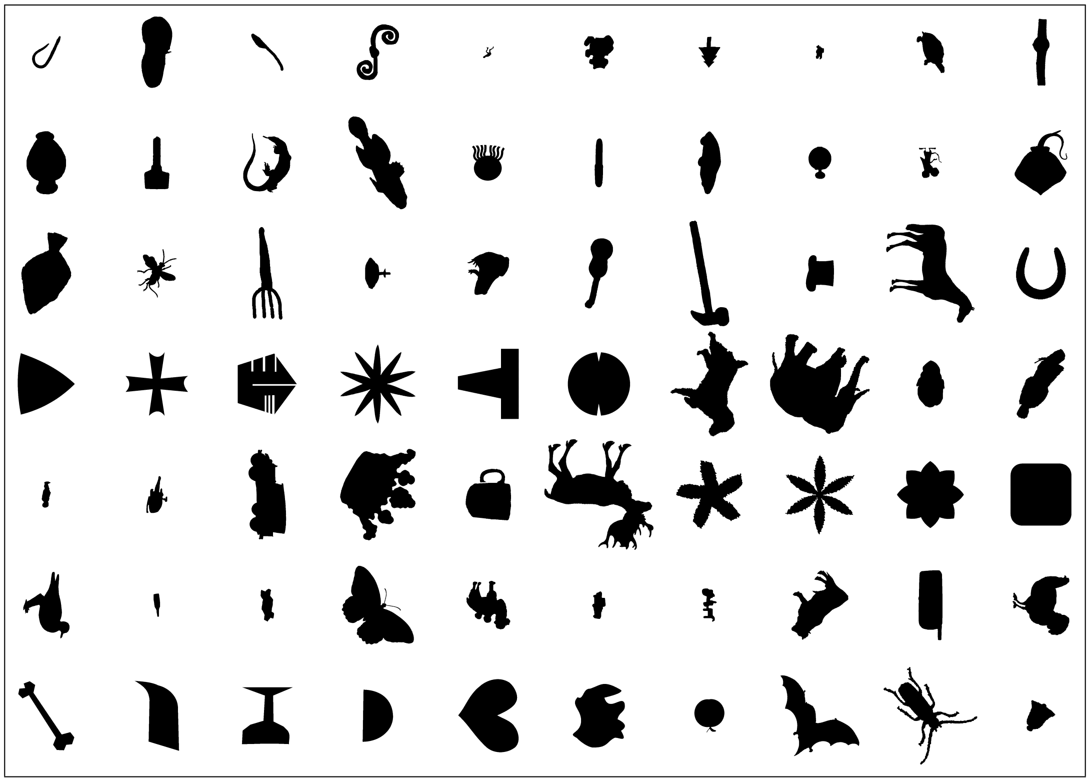
Figure 11: Some images in MPEG-7 CE-Shape-1 Part B database.

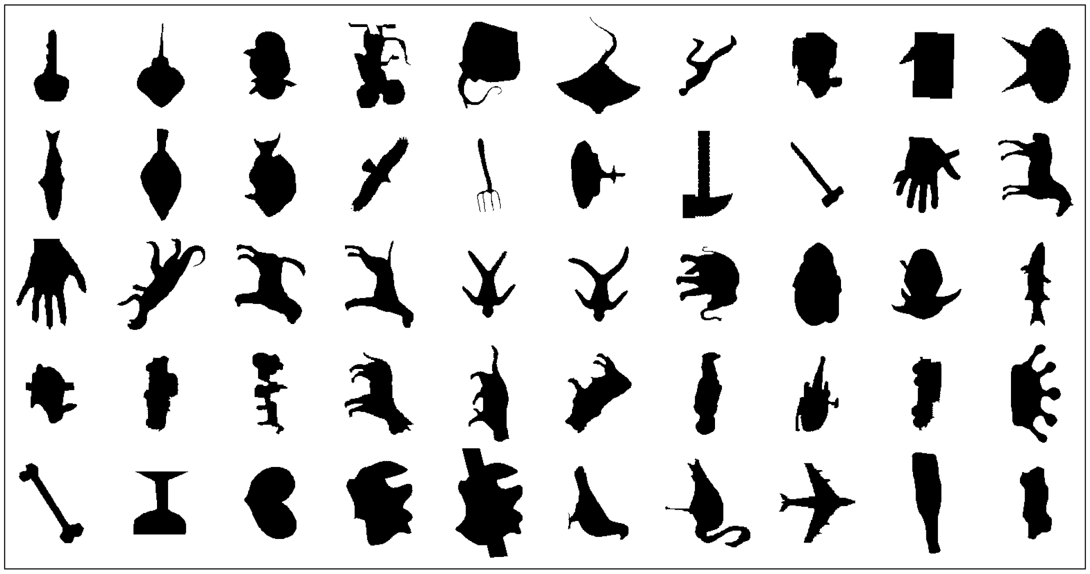
Figure 12: Some images in Silhouette database.

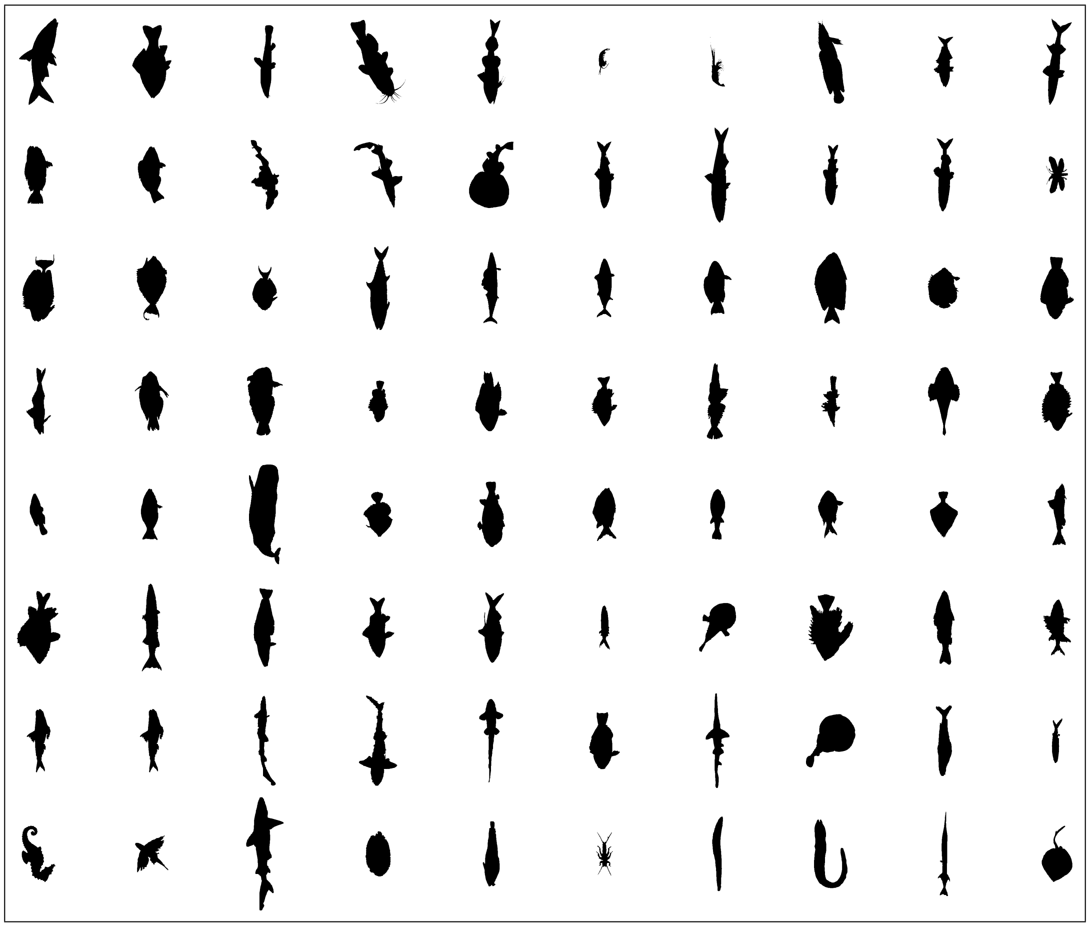
Figure 13: Some images in SQUID database.

- Beam angle statistics (BAS) [35]. A shape descriptor based on the curvature which in spite of having only four dimensions, its results are very competitive. For each point all the k-curvatures are obtained and four central moments are computed.

- Shape contexts [11]. For each point, a histogram of the relative coordinates of the remaining points is computed, where bins are uniform divisions of log-polar space centred at that point. As in [11], we used five distance bins and 12 angle bins, 60 dimensions.

We chose these shape descriptors to compare our proposal, CDTW, in the following ways. First, to see the improvement of our algorithm in shape retrieval rates with respect to canonical methods and also in time with respect to a brute force cyclic alignment and the canonical methods. These shape descriptors are representative examples from the literature and their results can be extrapolated to other similar methods [12, 13, 14, 15, 16], specially the results with the shape contexts [11]. Second, for measuring the behaviour of the dimensionality, as we say in Section 5, we have to take into account that it is not the same to compute a local distance between elements of one dimension (curvature) and to compute it between elements of 60 dimensions (shape contexts).

With regard to DTW and CDTW, $\omega = (1, 1, 1)$ was used and the local distance for the curvature was $\delta(a_i, b_j) = \sqrt{|a_i - b_j|}$, being a_i and b_j two curvature values, the same local distance for BAS but for a vector of four dimensions, and for the shape contexts $\delta(a_i, b_j) = \chi^2(a_i, b_j)$, being a_i and b_j vectors of 60 dimensions. The invariance to the mirror transformation was obtained by computing another DTW or CDTW with the shape descriptor of the mirror shape.

In the following we present the shape retrieval results in terms of rates and time. All of them has been obtained with leaving-one-out and an exhaustive search in corpuses.

6.1. Retrieval Rates

In Section 3 we comment two representative canonical methods to solve the problem of the invariance to the starting point: least second moment of inertia [7, 8] and Fourier Descriptors [9] (FDs). In this section we compare them with our proposal, in other words, we compare these heuristics with DTW against CDTW. As a base experiment we also show results with DTW without a method to obtain this invariance.

Performance of methods in the shape retrieval task is measured by:

- The bullseye test [2] (only for the MPEG-7 database). It consists in comparing each shape to every other shapes in the database. The retrieval rate for the query object is measured by counting the number of objects from the same category which are found in the first 40 most similar shapes.

- Mean Average Precision (MAP) [36]. For a set of images, MAP approximates the area under the Precision-Recall curve [37]. It has demonstrated to be a measure of quality, with a good discrimination and stability.

Table 1 shows the bullseye test results for the MPEG-7B corpus and Table 2 shows the MAP results, in this case for the three corpuses. CDTW has the best results for all the shape descriptors and corpuses.

Finally, to comment that if we use a context-sensitive learning approach, in particular [19], the bullseye test for the shape contexts and CDTW arrives to 94.15.

Bullseye test

Descriptor	**Method**	**Corpus**
		MPEG-7B
Curvature	DTW	46.16
	Moments+DTW	52.22
	FDs+DTW	58.03
	CDTW	**66.70**
BAS	DTW	56.57
	Moments+DTW	80.85
	FDs+DTW	80.97
	CDTW	**82.85**
Shape contexts	DTW	65.66
	Moments+DTW	84.65
	FDs+DTW	85.05
	CDTW	**86.73**

Table 1: Bullseye test to compare methods for obtaining the starting point with several shape descriptors. Bold entries show the best results.

MAP

Descriptor	**Method**	**Corpus**		
		MPEG-7B	Silhouette	SQUID
Curvature	DTW	36.30	33.54	22.34
	Moments+DTW	42.34	41.86	20.45
	FDs+DTW	49.15	45.18	28.89
	CDTW	**58.54**	**59.99**	**36.76**
BAS	DTW	48.46	41.57	31.19
	Moments+DTW	75.57	72.54	49.30
	FDs+DTW	75.64	72.61	50.15
	CDTW	**77.51**	**74.74**	**50.55**
Shape contexts	DTW	57.58	55.59	40.94
	Moments+DTW	80.01	80.99	54.80
	FDs+DTW	80.85	81.54	55.65
	CDTW	**82.63**	**83.97**	**56.12**

Table 2: MAP to compare methods for obtaining the starting point with several corpuses and shape descriptors. Bold entries show the best results.

6.2. Time

In this section, we also present time experiments. Times were measured on a 2.66GHz Intel i7 running under Linux 2.6.32. The algorithms were implemented in C++ and we used the GNU g++ compiler with -O2 optimizations.

Figures 14, 15 and 16 show the total relative time of: FDs+DTW (using FDs for obtaining the starting point, but it can be applied to any canonical method with similar properties) and CDTW, with respect to the brute force cyclic alignment time. The total times shown are for an exhaustive search for all the shapes in the corpuses.

For all the shape descriptors the speed up of CDTW with respect to brute force is very high. In the case of the shape contexts this speed up is a little lower due to the high cost over the computational time of the matrix of local distances. We have to calculate a χ^2 distance for each element of 60 dimensions in the matrix $n \times n$. To get a general idea, comparing with CDTW a query shape against the 1399 remaining samples (MPEG-7B), takes an average time of 2 seconds for the curvature, 2.2 seconds for BAS and 5.5 seconds for the shape contexts, approximately. In the case of the brute force cyclic alignment and shape contexts this time is prohibitive, 16.3 seconds.

Because of the dimensionality, [12, 13, 14, 15, 16] will also have prohibitive times. CDTW will significantly improve these times, as it happens with the shape contexts.

Regarding DTW and the canonical methods there is little improvement in time with respect to CDTW.

7. Discussion and Future Work

In this work, we have studied the brute force cyclic alignment and canonical methods detecting their drawbacks. As a better alternative we have defined the CDTW dissimilarity and an algorithm to compute it in $O(n^2 \log n)$ for two sequences of length n has been presented.

It has been shown that the CED algorithm cannot be directly extended to CDTW: two conventional DTW dissimilarities must be computed for each cyclic shift of one sequence. Fortunately, one of these dissimilarities can be obtained as a subproduct of the computation of the other. Thus a divide-and-conquer procedure is possible. Based on these problems, we have also defined extensions for global restrictions, other weights in the arcs and normalization.

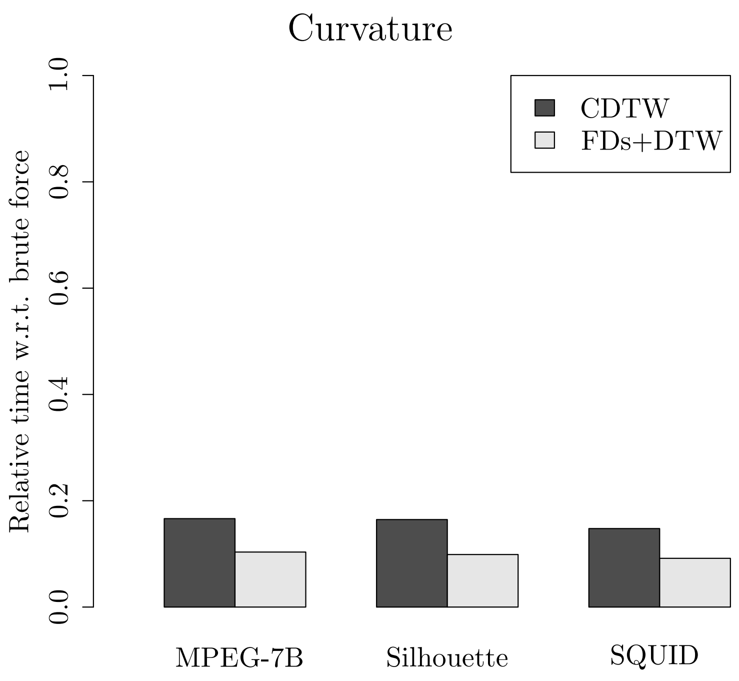

Figure 14: Relative total time comparison with respect to brute force cyclic alignment for FDs+DTW and CDTW methods with the curvature descriptor and several corpuses.

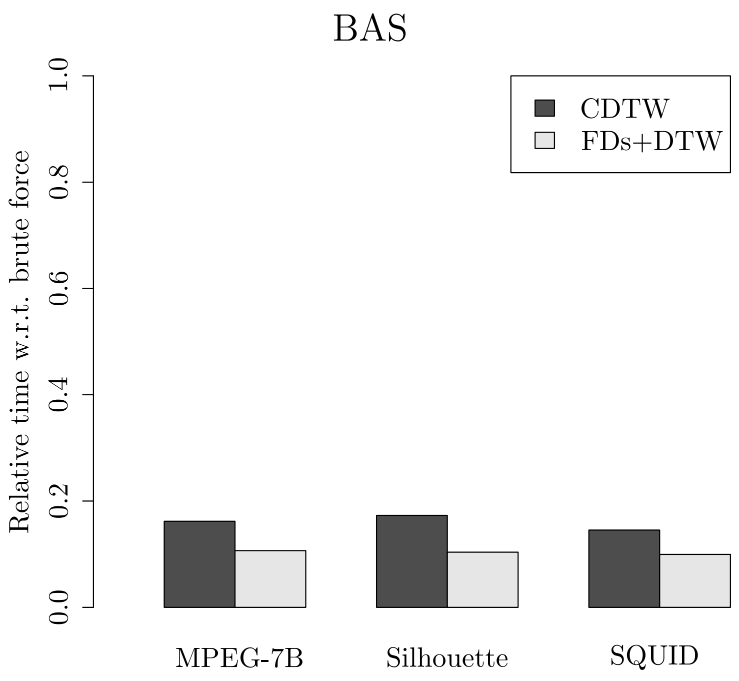

Figure 15: Relative total time comparison with respect to brute force cyclic alignment for FDs+DTW and CDTW methods with the BAS descriptor and several corpuses.

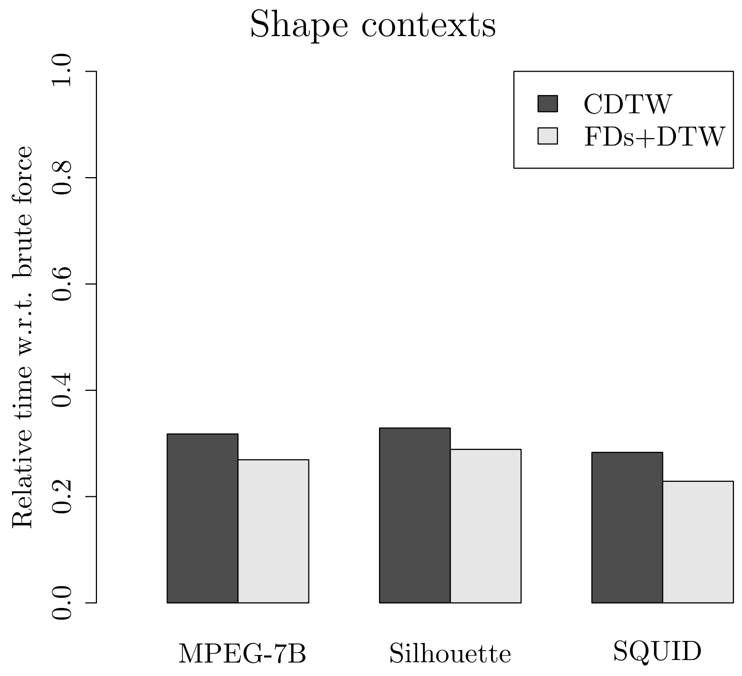

Figure 16: Relative total time comparison with respect to brute force cyclic alignment for FDs+DTW and CDTW methods with the shape contexts descriptor and several corpuses.

Experiments which were performed on different corpuses show that our proposal drastically speeds up the computation time with respect to the brute force, obviously, offering the same shape retrieval rates. Experiments also show that canonical methods do not offer a significant speed-up with respect to CDTW, in order to sacrifice this correct cyclic alignment.

Our algorithm can be used to significantly speed up the current state-of-the-art shape retrieval [11, 13, 14, 15, 16, 17, 18, 19].

In posterior work we aim to explore indexing methods with CDTW.

Appendix A. Other Weights in the Arcs: $\boldsymbol{\omega} = (1,1,1)$ and $\boldsymbol{\omega} \neq (1,1,1)$

In Theorem 1, we exposed that there are only two possible *cuts* in the alignment with $\omega = (1,1,1)$, either we cut one of the segments of B (when we choose the cyclic shift) or we do not cut it and the sequence is "cleanly" cut. In the case of other values of ω, there are two more possible cuts. In Figure A.17 there is an alignment that is also in Figure 2, but now with $\omega = (1,2,1)$. As we can see in Figure A.17a, there is a difference that is the alignment between a_3 and b_2. It corresponds to a right angle in the alignment path (see Figure A.17b).

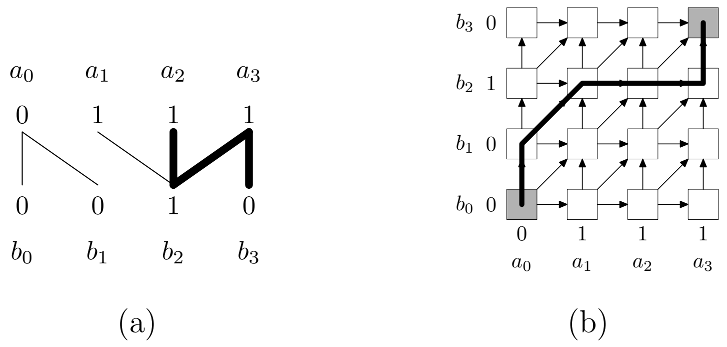

Figure A.17: (a) An optimal alignment between the sequences $A = 0111$ and $B = 0010$, where $\delta(a_i, b_j) = |a_i - b_j|$ and $\omega = (1, 2, 1)$. A thick stroke is marking the new type of alignment. (b) Warping graph. The thick line represents the optimal alignment that corresponds to (a).

Now, Lemma 2 is not followed (observe that $\omega \neq (1, 1, 1)$). In this case, the alignment includes $(2, 2)$, $(3, 2)$ and $(3, 3)$, but we cannot remove $(3, 2)$ to obtain one of lower cost. If we cut B between b_2 and b_3, one of these new types of cuts is produced. From now on, we call them N or И *cut* (for the similarities with the capital letter, and its mirror). As we can see in Figure A.17a, in the alignment $(2, 2)$, $(3, 2)$ and $(3, 3)$ there is an И cut. For instance, there would be an N cut if $(2, 2)$, $(2, 3)$ and $(3, 3)$ were included in the alignment (the other right angle in the warping graph, Figure A.17b).

These new cuts also invalidate Lemma 3, when $\omega \neq (1, 1, 1)$, because shifting the sequences for all the starting points does not provide a valid solution, as the following counter-example shows. Suppose that we want to compute $DTW_\omega([02], [131])$, being $\delta(a_i, b_j) = |a_i - b_j|$ and $\omega = (1, 3, 1)$. If we cyclically shift both sequences: $DTW_\omega(02, 113) = DTW_\omega(02, 131) = DTW_\omega(20, 131) = DTW_\omega(20, 311) = 4$ and $DTW_\omega(02, 311) = DTW_\omega(20, 113) = 6$. But the correct result for $DTW_\omega([02], [131])$ is 5 as Figure A.18 shows. In Figure A.18a the optimum alignment that is supposedly the result of the cyclic alignment is depicted, however, this is not so because we do not take into account weights in (3) for $i = 0$ and $j = 0$. a_0 should appear aligned with b_2 as in Figure A.18b.

Nevertheless, to achieve this result we must consider how we leave the graph at node $(m - 1, n - 1)$ to enter at node $(0, 0)$, in other words, we need to know with which weight we leave it. This leads us to find another expression for the DTW. From now on we call this new expression DTW2, because we use it to express the brute force with a cyclic shift of both sequences.

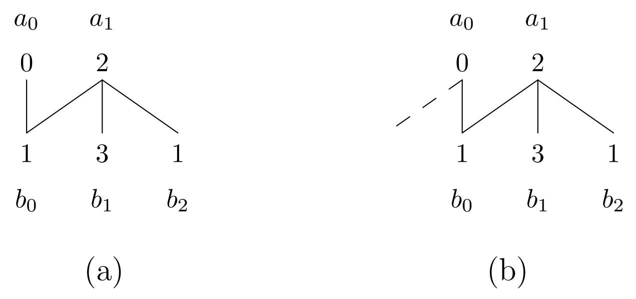

Figure A.18: (a) Optimal alignment between the sequences $A = 02$ and $B = 131$, where $\delta(a_i, b_j) = |a_i - b_j|$ and $\omega = (1, 3, 1)$. (b) Optimal alignment between the cyclic sequences $A = [02]$ and $B = [131]$. Dotted line indicates the alignment that is not in (a) for being a correct cyclic alignment.

Lemma 4. *The recursive equation for $DTW2_\omega(A, B) = d(m, n)$ is:*

$$d(i,j) = \begin{cases} 0, & \text{if } i = j = 0; \\ d(i-1,0) + w(1,0)\cdot\delta(a_i,b_0), & \text{if } i > 0 \text{ and } j = 0; \\ d(0,j-1) + w(0,1)\cdot\delta(a_0,b_j), & \text{if } i = 0 \text{ and } j > 0; \\ d(i-1,j-1) + & \\ \min\left\{ \begin{array}{l} w(1,1)\cdot\delta(a_0,b_0), \\ w(0,1)\cdot\delta(a_0,b_{m-1}) + w(1,0)\cdot\delta(a_0,b_0) \end{array} \right\}, & \text{if } i = m \text{ and } j = n; \\ \min\left\{ \begin{array}{l} d(i-1,j-1) + w(1,1)\cdot\delta(a_i,b_j), \\ d(i-1,j) + w(1,0)\cdot\delta(a_i,b_j), \\ d(i,j-1) + w(0,1)\cdot\delta(a_i,b_j) \end{array} \right\}, & \text{otherwise.} \end{cases} \tag{A.1}$$

CDTW for arbitrary weights between a cyclic sequence $[A]$ with size m and a cyclic sequence $[B]$ with size n, $CDTW_\omega([A],[B])$, can be computed with:

$$CDTW_\omega([A],[B]) = \min_{0 \le k < m} \left(\min_{0 \le \ell < n} DTW2_\omega(\sigma^k(A), \sigma^\ell(B)) \right). \tag{A.2}$$

Proof: We can take into account N and И cuts with the differences of (A.1) with respect to (3): (i) the case where $i = m$ and $j = n$, with it we obtain the suitable weight to enter at node $(0, 0)$; and (ii) the case where $i = j = 0$ that has as result 0 because we do not need to add here the local distance that was added in the case (i). However, in the case (i) we only consider И cuts (although we could take N cuts), because it is not necessary to consider the other cut, as we can observe in Figure A.19, for another of the cyclic alignments we can obtain an equivalent one. $\square$

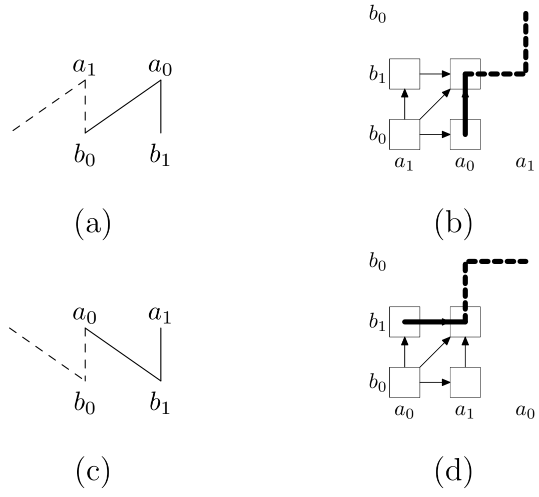

Figure A.19: Example of alignment with N and И cuts. Although we are only considering N or И in each case, alignments are equivalent. In (a) and (b), dotted lines correspond to the case where $i = m$ and $j = n$ in (A.1), in (c) and (d) they would also correspond to this case, supposing we had just considered the N cut instead of the И cut. Although we begin at (0,0), line does not appear in this node to indicate that there is a value 0 at the beginning. (a) Alignment where there is an И cut. (b) Warping graph that corresponds to the alignment in (a). (c) Alignment where there is an N cut. (d) Warping graph that corresponds to the alignment in (c).

According to Lemma 4, $CDTW_{\omega}([A],[B])$ can be computed in $O(m^2n^2)$ time with all the cyclic shifts of both sequences. But we need to compute it with the cyclic shift of only one of the sequences. Thus, we have to define DTW1.

Theorem 5. *The recursive equation for $DTW1_\omega(A,B) = d(m,n)$ is:*

$$d(i,j) = \begin{cases} 0, & \text{if } i = j = 0; \\ d(i-1,0) + w(1,0)\cdot\delta(a_i,b_0), & \text{if } i > 0 \text{ and } j = 0; \\ d(0,j-1) + w(0,1)\cdot\delta(a_0,b_j), & \text{if } i = 0 \text{ and } j > 0; \\ \min\left\{\begin{array}{l} d(i-1,j-1) + w(1,1)\cdot\delta(a_0,b_0), \\ d(i,j-1) + w(1,0)\cdot\delta(a_0,b_0) \end{array}\right\}, & \text{if } i = m \text{ and } j = n; \\ \min\left\{\begin{array}{l} d(i-1,j-1) + w(1,1)\cdot\delta(a_0,b_j), \\ d(i-1,j) + w(1,0)\cdot\delta(a_0,b_j), \\ d(i,j-1) + w(0,1)\cdot\delta(a_0,b_j) \end{array}\right\}, & \text{if } i = m \text{ and } j \neq n; \\ \min\left\{\begin{array}{l} d(i-1,j-1) + w(1,1)\cdot\delta(a_i,b_j), \\ d(i-1,j) + w(1,0)\cdot\delta(a_i,b_j), \\ d(i,j-1) + w(0,1)\cdot\delta(a_i,b_j) \end{array}\right\}, & \text{otherwise.} \end{cases} \tag{A.3}$$

CDTW for arbitrary weights between a cyclic sequence $[A]$ and a cyclic sequence $[B]$ can be computed with:

$$CDTW_\omega([A],[B]) = \min_{0\le k<m} DTW1_\omega(\sigma^k(A),B). \tag{A.4}$$

Proof: In (A.3), (7) and (A.1) are unified. On one hand, in the case $i = m$ and $j \neq n$, the concatenation with the first element of the sequence is done (see (7)), that is to say, there is a new column in the graph (see Figure 7c). On the other hand, in the case $i = m$ and $j = n$, how we enter at node $(0,0)$ is considered. $\square$

460 In Figure A.20 we can see the example of Figure A.18 with the solution using Lemma 4 and Theorem 5.

The extension of the Divide-and-Conquer procedure with CDTW using arbitrary weights is not complicated. We only translate to the extended graph the specific cases in which N and И cuts are treated (A.3), the case $i = j = 0$ and the case $i = m$ and $j = n$, for each of the cyclic shifts.
465 In Figure A.21, an example is depicted.

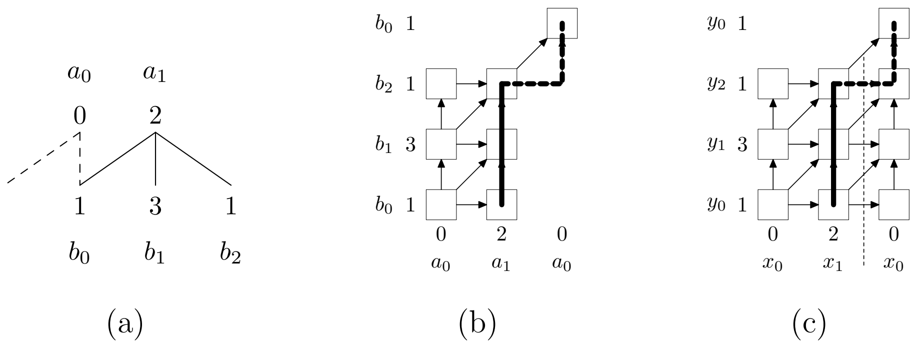

Figure A.20: (a) Optimal alignment between cyclic sequences $A = [02]$ and $B = [131]$, where $\delta(a_i, b_j) = |a_i - b_j|$ and $\omega = (1, 3, 1)$. (b) Warping graph corresponding to (a) that is obtained using (A.1). The graph corresponds to the cyclic minimization of both sequences. Although we begin at (0,0), line does not appear in this node to indicate that there is a value 0 at the beginning, as (A.1) indicates. (c) Warping graph corresponding to (a) that is obtained using (A.3). The graph corresponds to the cyclic minimization of the sequence A.

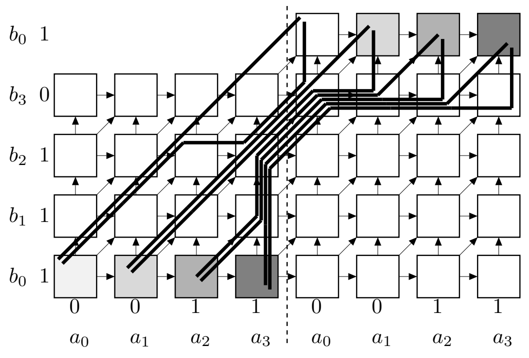

Figure A.21: Extended warping graph for arbitrary weights. With the cyclic sequences $A = [0011]$ and $B = [1110]$, where $\delta(a_i, b_j) = |a_i - b_j|$ and $\omega = (1, 1, 1)$. The optimal alignment for $[A]$ and $[B]$ is the path with minimum cost starting from any colored node in the lower row and ending at a node containing the same color in the upper row (all candidate paths are shown with thick lines).

Appendix B. Normalization

Following [26] and using DTW1 (see Appendix A), we can define the normalized CDTW (NCDTW), of $[A]$ and $[B]$ for arbitrary weights, ω, as:

$$NCDTW_\omega([A],[B]) = \min_{0\leq k<m} NDTW1_\omega(\sigma^k(A),B) = \min_{A\in[A]} \min_{p\in\mathcal{P}} \frac{W(p)}{L(p)}. \tag{B.1}$$

Thus, problems are:

Problem N.

$$d^* = \min_{A\in[A]} \min_{p\in\mathcal{P}} \frac{W(p)}{L(p)}, \quad W, L : \mathcal{P}; \quad L(p) > 0,\ \forall p \in \mathcal{P}. \tag{B.2}$$

Problem N can be solved by means of Problem $N(\lambda)$:

Problem N(λ).

$$d^*(\lambda) = \min_{A\in[A]} \min_{p\in\mathcal{P}} (W(p) - \lambda L(p)), \quad W, L : \mathcal{P}; \quad L(p) > 0,\ \forall p \in \mathcal{P}. \tag{B.3}$$

Using the Divide-and-Conquer procedure the computational time is $O(tmn\log(m))$, being t the number of iterations.

Acknowledgments

The authors wish to thank S. Abbasi, F. Mokhtarian, and J. Kittler for making the SQUID database publicly available and to I. Bartolini, P. Ciaccia, and M. Patella for providing their labeled version of the SQUID database.

Work partially supported by the Spanish Government (TIN2010-18958 and *Consolider Ingenio 2010* CSD2007-00018), the Generalitat Valenciana (*Prometeo*/2010/028), and *Fundació Caixa Castelló-Bancaixa* (P11B2009-48).

[1] T. Sikora, The MPEG-7 visual standard for content description – an overview, IEEE Transactions on Circuits and Systems for Video Technology 11 (6) (2001) 696–702.

[2] M. Bober, MPEG-7 visual shape descriptors, IEEE Transactions on Circuits and Systems for Video Technology 11 (6) (2001) 716–719.

[3] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, The MIT Press Cambridge, MA, 1990.

[4] R. Wagner, M. Fisher, The string-to-string correction problem, Journal of the ACM 21 (1) (1974) 168–173.

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

[6] E. Petrakis, A. Diplaros, E. Milios, Matching and retrieval of distorted and occluded shapes using dynamic programming, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (11) (2002) 1501–1516.

[7] B. Horn, Robot vision, The MIT Press, 1986.

[8] R. Jain, R. Kasturi, B. Schunck, Machine vision, Vol. 5, McGraw-Hill New York, 1995.

[9] I. Bartolini, P. Ciaccia, M. Patella, WARP: Accurate retrieval of shapes using phase of Fourier descriptors and time warping distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (1) (2005) 142–147.

[10] J. Crespo, P. Aguiar, Revisiting complex moments for 2d shape representation and image normalization, IEEE Transactions on Image Processing 20 (10) (2011) 2896–2911.

[11] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (4) (2002) 509–522.

[12] T. Adamek, N. E. O'Connor, A multiscale representation method for nonrigid shapes with a single closed contour, IEEE Transactions on Circuits and Systems for Video Technology 14 (5) (2004) 742–753.

[13] N. Alajlan, I. E. Rube, M. S. Kamel, G. Freeman, Shape retrieval using triangle-area representation and dynamic space warping, Pattern Recognition 40 (7) (2007) 1911–1920.

[14] H. Ling, D. W. Jacobs, Shape classification using the inner-distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2) (2007) 286–299.

[15] R. Gopalan, P. Turaga, R. Chellappa, Articulation-invariant representation of non-planar shapes, in: European Conference on Computer Vision, 2010, pp. 286–299.

[16] 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.

[17] X. Bai, X. Yang, L. Latecki, W. Liu, Z. Tu, Learning context-sensitive shape similarity by graph transduction, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (5) (2010) 861–874.

[18] X. Yang, S. Koknar-Tezel, L. Latecki, Locally constrained diffusion process on locally densified distance spaces with applications to shape retrieval, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 357–364.

[19] P. Kontschieder, M. Donoser, H. Bischof, Beyond pairwise shape similarity analysis, in: Asian Conference on Computer Vision, 2010, pp. 655–666.

[20] M. Maes, On a cyclic string-to-string correction problem, Information Processing Letters 35 (2) (1990) 73–78.

[21] E. Keogh, L. Wei, X. Xi, M. Vlachos, S. Lee, P. Protopapas, Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures, The VLDB Journal 18 (3) (2009) 611–630.

[22] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech and Signal Processing 26 (1) (1978) 43–49.

[23] C. Myers, L. Rabiner, A. Rosenberg, Performance tradeoffs in dynamic time warping algorithms for isolated word recognition, IEEE Transactions on Acoustics, Speech and Signal Processing 28 (6) (1980) 623–635.

[24] L. Rabiner, B.-H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, 1993.

[25] A. Marzal, E. Vidal, Computation of normalized edit distance and applications, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (9) (1993) 926–932.

[26] E. Vidal, A. Marzal, P. Aibar, Fast computation of normalized edit distances, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (9) (1995) 899–902.

[27] R. Klette, A. Rosenfeld, Digital geometry: geometric methods for digital picture analysis, Morgan Kaufmann, 2004.

[28] A. Folkers, H. Samet, Content-based image retrieval using Fourier descriptors on a logo database, in: International Conference on Pattern Recognition (3), 2002, pp. 521–524.

[29] A. Marzal, S. Barrachina, Speeding up the computation of the edit distance for cyclic strings, in: International Conference on Pattern Recognition, 2000, pp. 483–519.

[30] E. J. Keogh, C. A. Ratanamahatana, Exact indexing of dynamic time warping, Knowledge and Information Systems 7 (3) (2005) 358–386.

[31] L. Latecki, R. Lakämper, U. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: IEEE Conference on Computer Vision and Pattern Recognition, 2000, pp. 424–429.

[32] 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.

[33] F. Mokhtarian, J. Kittler, S. Abbasi, Shape queries using image databases, http://www.ee.surrey.ac.uk/Research/VSSP/imagedb/demo.html.

[34] F. Mokhtarian, A. K. Mackworth, A theory of multiscale, curvature-based shape representation for planar curves, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (8) (1992) 789–805.

[35] N. Arica, F. T. Yarman-Vural, BAS: a perceptual shape descriptor based on the beam angle statistics, Pattern Recognition Letters 24 (9-10) (2003) 1627–1639.

[36] C. D. Manning, P. Raghavan, H. Schtze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.

[37] C. J. Rijsbergen, Information Retrieval, London: Butterworth, 1979.