JOURNAL OF THE ROYAL SOCIETY Interface

# Supporting anthropological research with efficient rotation invariant shape similarity measurement

L Wei, E Keogh, X Xi and S.-H Lee

*J. R. Soc. Interface* 2007 **4**, 207-222
doi: 10.1098/rsif.2006.0168

JOURNAL OF THE ROYAL SOCIETY Interface

*J. R. Soc. Interface* (2007) **4**, 207–222
doi:10.1098/rsif.2006.0168
*Published online* 17 October 2006

# Supporting anthropological research with efficient rotation invariant shape similarity measurement

**L. Wei[1], E. Keogh[1,*], X. Xi[1] and S.-H. Lee[2]**

[1]*Department of Computer Science & Engineering, and* [2]*Department of Anthropology, University of California-Riverside, Riverside, CA 92521, USA*

The matching of two-dimensional shapes is an important problem with many applications in anthropology. Examples of objects that anthropologists are interested in classifying, clustering and indexing based on shape include bone fragments, projectile points (arrowheads/spearpoints), petroglyphs and ceramics. Interest in matching such objects originates from the fundamental question for many biological anthropologists and archaeologists: how can we best quantify differences and similarities? This interest is fuelled in part by a movement that notes: 'an increasing number of archaeologists are showing interest in employing Darwinian evolutionary theory to explain variation in the material record'. Aiding such research efforts with computers requires a shape similarity measure that is invariant to many distortions, including scale, offset, noise, partial occlusion, etc. Most of these distortions are relatively easy to handle, either in the representation of the data or in the similarity measure used. However, rotation invariance seems to be uniquely difficult. Current approaches typically try to achieve rotation invariance in the representation of the data, at the expense of poor discrimination ability, or in the distance measure, at the expense of efficiency. In this work, we show that we can take the slow but accurate approaches and dramatically speed them up. On real world problems, our technique can take current approaches and make them four orders of magnitude faster, without false dismissals. Moreover, our technique can be used with *any* of the dozens of existing shape representations and with *all* the most popular distance measures, including Euclidean distance, dynamic time warping and longest common subsequence. We show the applications of our work to several important problems in anthropology, including clustering and indexing of skulls, projectile points and petroglyphs.

**Keywords: shape; indexing; rotation invariance; dynamic time warping; anthropology**

## 1. INTRODUCTION

Anthropologists often deal with physical (as opposed to purely social or linguistic, etc.) artefacts. While the colour or texture of such artefacts may be of interest, it is often the case that the *shape* is of most interest. Examples of artefacts that anthropologists are interested in classifying, clustering or indexing based on shape include bone fragments, projectile points (arrowheads/spearpoints), petroglyphs and ceramics. While anthropologists have long been interested in shape, interest in matching such objects is fuelled in part by the availability of computing power and by a recent movement that notes: 'an increasing number of archaeologists are showing interest in employing Darwinian evolutionary theory to explain variation in the material record' (O'Brien & Lyman 2003). For example, anthropologists have recently used tools from biological morphology to attempt to explain spatial and temporal distribution of projectile points in North America. Aiding such research efforts with computers requires a shape similarity measure that is invariant to many distortions, including scale, offset, noise, partial occlusion, etc. Most of these distortions are relatively easy to handle, particularly if we use the well-known technique of converting the shapes into time-series as in figure 1.

However, no matter what representation is used, rotation invariance seems to be uniquely difficult to handle. For example, Li & Simske (2002) notes that 'rotation is always something hard to handle compared with translation and scaling', and the literature abounds with similar statements. Many current approaches try to achieve rotation invariance in the representation of the data, at the expense of discrimination ability (Osada *et al.* 2002), or in the distance measure, at the expense of efficiency (Gdalyahu & Weinshall 1999; Adamek & O'Connor 2003, 2004; Attalla & Siy 2005).

As an example of the former, the very efficient rotation invariant technique of Osada *et al.* (2002) cannot differentiate between the shapes of the lowercase letters

*Author for correspondence (eamonn@cs.ucr.edu).

*Received* 7 August 2006
*Accepted* 21 September 2006

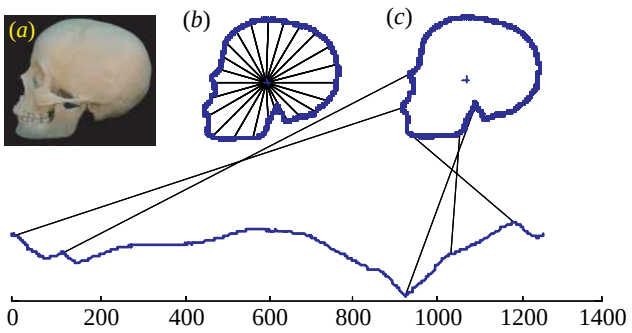


Figure 1. Shapes can be converted to time-series. (*a*) A bitmap of a human skull. (*b*) The distance from every point on the profile to the centre is measured and treated as the Y-axis of a time-series of length $n$, see (*c*).

'd' and 'b'. As an example of the latter, the work of Adamek & O'Connor (2004), which is the state of the art in terms of accuracy or precision/recall, takes an untenable amount of time for each shape comparison.

In this work,[1] we show that we can take the slow but accurate approaches and dramatically speed them up. For example, we can take the $O(n^3)$ approach of Adamek & O'Connor (2004) and on real world problems bring the average complexity down to $O(n^{1.06})$. This dramatic improvement in efficiency does not come at the expense of accuracy; the algorithm is guaranteed to return the same answer set as the slower methods.

We achieve speedup of the existing methods by dramatically decreasing the central processing unit (CPU) requirements. Our technique works by grouping together similar rotations and defining an admissible lower bound to that group. Given a tight and admissible lower bound, we can use the many search techniques known in the database community.

Our technique has the following advantages:

— there are dozens of techniques in the literature for converting shapes to time-series (Wang *et al.* 2000; Cardone *et al.* 2003; Adamek & O'Connor 2004; Zhang & Lu 2004; Attalla & Siy 2005; Vlachos *et al.* 2005), including some that are domain specific (Rath & Manmatha 2002; Bhanu & Zhou 2004). Our approach works for *any* of these representations.
— while there are many distance measures for shapes in the literature, Euclidean distance, dynamic time warping (Rath & Manmatha 2002; Adamek & O'Connor 2003; Bhanu & Zhou 2004; Ratanamahatana & Keogh 2005) and longest common subsequence (Vlachos *et al.* 2005) account for the majority of the literature. Our approach works for *any* of these distance measures.
— our approach uses the idea of envelope lower bounding as its cornerstone. Since the introduction of this idea a few years ago (Keogh 2002), dozens of researchers worldwide have adopted and extended this framework for applications as diverse as motion capture indexing (Keogh & Kasetty 2002), P2P searching (Karydis *et al.* 2005), handwriting retrieval (Rath & Manmatha 2002), dance indexing, query by humming and monitoring streams (Wei *et al.* 2005). This widespread adoption of envelope lower bounding has insured that it has become a mature and widely supported technology, and it suggests that any contributions made here can be rapidly adopted and expanded.

The rest of this paper is organized as follows. In §2, we discuss the background material and the related work. In §3, we formally introduce the problem and in §4, we offer our solution. Section 5 offers a comprehensive empirical evaluation of our technique. Finally, §6 offers some conclusions and directions for future work.

## 2. BACKGROUND AND RELATED WORK

The literature on shape matching is vast; we refer the interested reader to Cardone *et al.* (2003), Zhang & Lu (2004) and Veltkamp & Latecki (2006) for excellent surveys. While not all work on shape matching uses a one-dimensional representation of the two-dimensional shapes, an increasingly large majority of the literature does. Therefore, we consider only such approaches here. Note that we lose little by this omission. The two most popular measures that operate directly in the image space, the chamfer (Borgefors 1988) and Hausdorff (Olson & Huttenlocher 1997) distance measures, require $O(n^2 \log n)$ time,[2] and recent experiments (including some in this work) suggest that one-dimensional representations can achieve comparable or superior accuracy.

In essence, there are three major techniques for dealing with rotation invariance, landmarking, rotation invariant features and brute force rotation alignment. We consider each in §§§2.1–2.3.

### *2.1. Landmarking*

The idea of 'landmarking' is to find the one 'true' rotation and only use that particular alignment as the input to the distance measure. The idea comes in two flavours: domain dependent and domain independent.

In domain dependent landmarking, we attempt to find a single (or very few) fixed feature to use as a starting point for the conversion of the shape to a time-series. For example, in face profile recognition, the most commonly used landmarks (fiducial points) are the chin or the nose (Bhanu & Zhou 2004). In limited domains, this may be useful, but it requires building special-purpose feature extractors. For example, even in a domain as intuitively well understood as human profiles, accurately locating the nose is a non-trivial problem, even if we discount the possibility of mustaches and glasses. Probably, the only reason any progress has been made in this area is that most work

[1]In this work, we make use of Big $O$ notation such as '$O(n^3)$'. See appendix A for an intuition for this notation.

[2]More precisely, the time complexity is $O(Rp \log p)$, where $p$ is the number of pixels in the perimeter and $R$ is the number of rotations that need to be executed. Here $p=n$, and while $R$ is a user-defined parameter, it should be approximately equal $n$ to guarantee all rotations (up to the limit of rasterization) are considered.

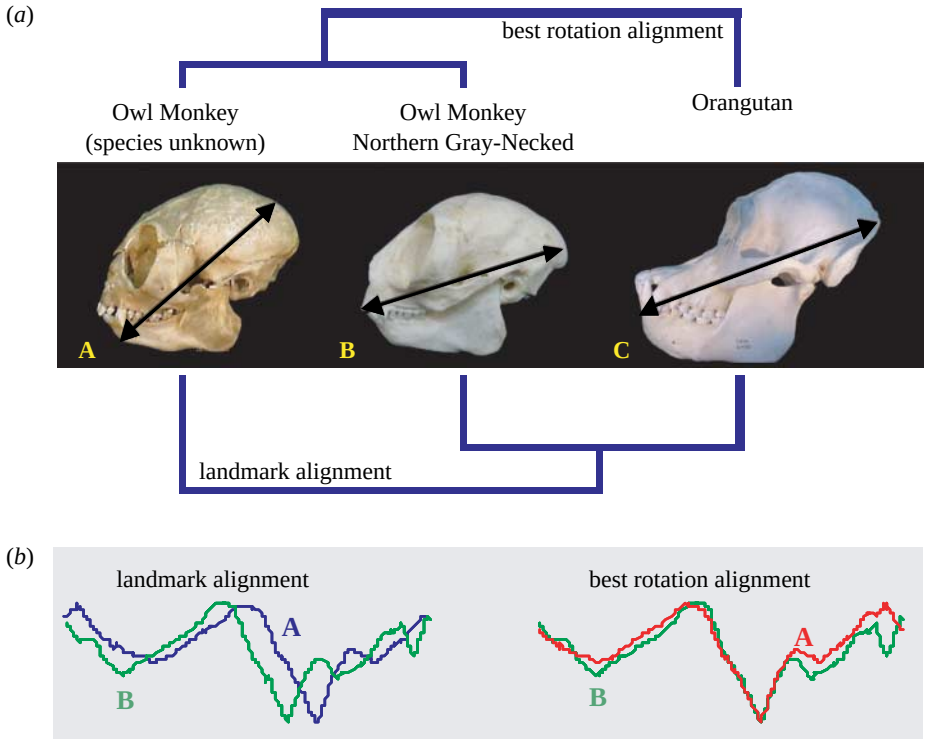


Figure 2. (*a*) Three primate skulls, two of them from the same genus, are clustered using both the landmark rotation beginning at the major axis and the best rotation. (*b*) The landmark-based alignment of A and B explains why the landmark-based clustering is incorrect: a small amount of rotation error results in a large difference in the distance measure.

reasonably assumes that faces presented in an image are likely to be upright. For shape matching in skulls, the canonical landmark is called the Frankfurt horizontal (White 2000), which is defined by the right and left *porion* (the highest point on the margin of the external auditory meatus) and the left *orbitale* (the lowest point on the orbital margin). However, a skull can be missing the relevant bones to determine this orientation and still have enough global information to match its shape to similar examples. Indeed, the Skhul V skull shown in figure 12 is such an example.

In domain independent landmarking, we align all the shapes to some cardinal orientation, typically the major axis. This approach may be useful for the limited domains in which there is a well-defined major axis, perhaps the indexing of hand tools. However, there is an increasing recognition that the '…major axis is sensitive to noise and unreliable' (Zhang & Lu 2004). For example, a recent paper shows that under some circumstances, a single extra pixel can change the rotation by $\pm 90°$ (Zunic *et al.* 2006).

To show how brittle landmarking can be, we performed a simple clustering experiment where we clustered three primate skulls using Euclidean distance, with both the major axis technique and the minimum distance of all possible rotations (as found by brute force). Figure 2 shows the result. It is clear that the major axes do not have any biological meaning: the points connecting each axis for each specimen are not homologous (of shared evolutionary origin). Therefore, the resulting clusters are meaningless in terms of biology and morphology.

The most important lesson we learned from this experiment (and dozens of other similar experiments on diverse domains; see Keogh 2006) is that rotation (mis)alignment is the most important invariance for shape matching; unless we have the best rotation, then nothing else matters.

### 2.2. Rotation invariant features

A large number of papers achieve fast rotation invariant matching by extracting only the rotation invariant features and indexing them with a feature vector (Cardone *et al.* 2003). This feature vector is often called the shapes 'signature'. There are literally dozens of rotation invariant features, including ratio of perimeter to area, fractal measures, elongatedness, circularity, min/max/mean curvature, entropy, perimeter of convex hull, etc. In addition, many researchers have attempted to frame the shape-matching problem as a more familiar histogram-matching problem. For example, in Osada *et al.* 2002, the authors build a histogram containing the distances between two randomly chosen points on the perimeter of the shapes in question. The approach seems to be attractive, for example, it can trivially also handle three-dimensional shapes, however, it suffers from extremely poor precision. For example, it cannot differentiate between the shapes of the lowercase letters 'd' and 'b' or 'p' and 'q', since these pairs of shapes have identical histograms. In general, all these methods suffer from very poor discrimination ability (Cardone *et al.* 2003).

### 2.3. Brute force rotation alignment

There are a handful of papers that recognize that the above attempts at approximating rotation invariance are unsatisfactory for most domains, and they achieve

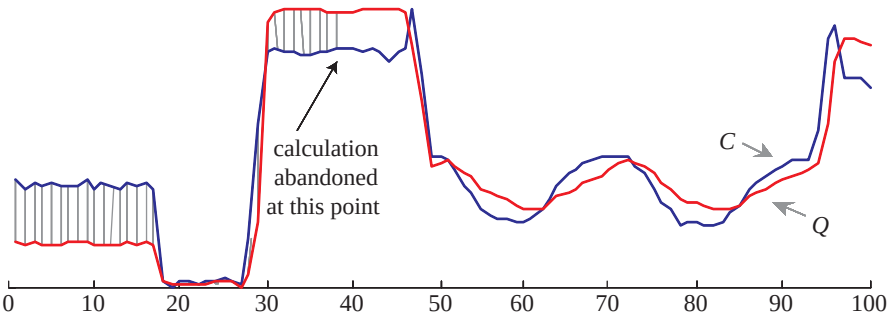


Figure 3. A visual intuition of early abandoning. Once the squared sum of the accumulated grey hatch lines exceeds $r^2$, we can be sure that the full Euclidean distance exceeds $r$.

true rotation invariance by an exhaustive brute force search over all possible rotations, but only at the expense of computational efficiency (Gdalyahu & Weinshall 1999; Wang *et al.* 2000; Adamek & O'Connor 2003, 2004; Attalla & Siy 2005). For example, the paper of Adamek & O'Connor (2004) uses dynamic time warping (DTW is discussed in detail in §4.2) to handle non-rigid shapes in the time-series domain; while they note that most invariances are trivial to handle in this representation, they state that 'rotation invariance can (only) be obtained by checking all possible circular shifts for the optimal diagonal path'. This step makes the comparison of two shapes $O(n^3)$. Similarly, the paper of Wang *et al.* (2000) notes that 'in order to find the best matching result, we have to shift one curve $n$ times, where $n$ is the number of possible start points'.

## 3. ROTATION INVARIANT MATCHING

We begin by formally defining the rotation invariant matching problem and by assuming the Euclidean distance, and we generalize to other distance measures later. For clarity of presentation, we will generally refer to 'time-series', which the reader will note can be mapped back to the original shapes.

Suppose that we have two time-series, $Q$ and $C$ of length $n$, which were extracted from shapes by an arbitrary method,

$$Q = q_1, q_2, \ldots, q_i, \ldots, q_n,$$
$$C = c_1, c_2, \ldots, c_j, \ldots, c_n.$$

As we are interested in large data collections, we denote a database of $m$ such time-series as $\bar{Q}$,

$$\bar{Q} = \{Q_1, Q_2, \ldots, Q_m\}.$$

If we wish to compare two time-series, and therefore shapes, we can use the ubiquitous Euclidean distance

$$\mathrm{ED}(Q, C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}.$$

When using the Euclidean distance as a subroutine in a classification or indexing algorithm, we may be interested in knowing the exact distance only when it is eventually going to be less than some threshold $r$. For example, this threshold can be the 'range' in range search or the 'best-so-far' in nearest neighbour search. If this is the case, we can potentially speedup the calculation by doing early abandoning (Agrawal *et al.* 1993; Keogh & Kasetty 2002).

Table 1. Euclidean distance optimized with early abandonment.

```
algorithm [dist, num_steps]=EA_Euclidean_Dist(Q, C, r)
accumulator=0
for i=1 to length(Q)                    //loop over time-series
  accumulator+=(q_i-c_i)^2              //accumulate error contribution
  if accumulator>r^2                    //can we abandon?
    disp('doing an early abandon')
    num_steps=i
    return [infinity, num_steps]        //terminate and return an
  end                                   //infinite error to signal the
end                                     //early abandonment.
return [sqrt(accumulator),              //terminate with true dist
    length(Q)]
```

**Definition 3.1.** *Early abandon.* During the computation of the Euclidean distance, if we note that the current sum of the squared differences between each pair of corresponding data points exceeds $r^2$, then we can stop the calculation, secure in the knowledge that the exact Euclidean distance had we calculated it would exceed $r$.

While the idea of early abandoning is fairly obvious and intuitive, it is so important to our work that we illustrate it in figure 3 and provide pseudocode in table 1.

Note that the 'num_steps' value returned by the optimized Euclidean distance in table 1 is used only to tell us how useful the optimization was. If its value is significantly less than $n$, then this suggests a dramatic speedup.

While the Euclidean distance is a simple distance measure, it produces surprisingly good results for clustering, classification and query by content of shapes, *if* the time-series in question happen to be rotation aligned. For example, in an experiment in Ratanamahatana & Keogh (2005), we manually performed rotation alignment of the time-series extracted from face profiles by explicitly showing the algorithm the beginning and the endpoint of a face (the nape and the Adam's apple, respectively).

However, if the shapes are not rotation aligned, this method can produce extremely poor results. To overcome this problem, we need to hold one shape fixed,

rotate the other and record the minimum distance of all possible rotations.

For reasons that will become apparent later, we achieve this by expanding one time-series into a matrix $\boldsymbol{C}$ of size $n$ by $n$,

$$\boldsymbol{C} = \begin{Bmatrix} c_1, c_2, \ldots, c_{n-1}, c_n \\ c_2, \ldots, c_{n-1}, c_n, c_1 \\ \vdots \\ c_n, c_1, c_2, \ldots, c_{n-1} \end{Bmatrix}.$$

Note that each row of the matrix is simply a time-series, shifted (rotated) by one from its neighbours. It will be useful below to address the time-series in each row individually, so we will denote the $i$th row as $C_i$, which allows us to denote the matrix above in the more compact form of $\boldsymbol{C}=\{C_1, C_2, \ldots, C_n\}$.

We can now define the rotation invariant Euclidean distance (RED) as

$$\mathrm{RED}(Q, \boldsymbol{C}) = \min_{1\le j\le n} \left\{ \mathrm{ED}(Q, C_j) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}. \right.$$

Table 2 shows the pseudocode to calculate this.

Note that the algorithm tries to take advantage of early abandoning by passing EA_Euclidean_Dist the value of $r$, the best rotation alignment discovered thus far.

If we are simply measuring the distance between two time-series, then the algorithm is invoked, with $r$ set to infinity; however, as we shall see below, if the algorithm is being used as a subroutine in a linear scan of a large dataset $\bar{Q}$, the calling routine can set the value of $r$ to achieve speedup. In particular, the calling function sets $r$ to the value of the best match (under any rotation) discovered thus far. Table 3 shows the pseudocode. Note that the time complexity for this algorithm is $O(mn^2)$. This is simply untenable for large datasets.

Before continuing, we will review the notation introduced thus far in table 4.

Note that our notation seems somewhat space inefficient, in that it expands time-series $C$, of length $n$, to a matrix of size $n\times n$. However, the rest of the database uses the original (arbitrary rotation) time-series, and since the size of the database is assumed to be large, this overhead is asymptotically irrelevant.

Depending on the application, we may wish to retrieve the shapes that are enantiomorphic (mirror images) to the query. For example, in matching skulls, the best match may simply be facing the opposite direction. In contrast, when matching letters, we *do not* want to match a 'd' to a 'b'. If enantiomorphic invariance is required, we can trivially achieve this by augmenting matrix $\boldsymbol{C}$ to contain $C_i$ and reverse($C_i$), for $1\le i\le n$.

Thus far, we have shown a brute force search algorithm that can support rotation invariance, rotation-limited invariance and/or mirror image invariance. We simply put the appropriate time-series into matrix $\boldsymbol{C}$ and invoke the algorithm in table 3. This algorithm, even though speeded up by the early abandoning optimization, is too slow for large datasets. In §4, we introduce our novel search mechanism.

Table 2. An algorithm to find the rotated match between two time-series.

```
algorithm: [bestSoFar] = Test_All_Rotations(Q, C, r)
bestSoFar = r
for j = 1 to n
 distance = EA_Euclidean_Dist(Q, C_j, bestSoFar) //as in
   table 1
  if distance < bestSoFar
   bestSoFar = distance
  end
end
return [bestSoFar]
```

Table 3. An algorithm to find the best rotated match to query from a database of possible matches.

```
algorithm: [best_match_loc, bestSoFar] =
   Search_Database_for_Rotated_Match(C, Q̄)
best_match_loc = null
bestSoFar = inf
for i = 1 to number_of_time_series_in_database(Q̄)
 distance = Test_All_Rotations(Q̄, C, bestSoFar) //as in
   table 2
  if distance < bestSoFar
   best_match_loc = i
   bestSoFar = distance
  end
end
return [best_match_loc, bestSoFar]
```

Table 4. Notation table.

| | |
|---|---|
| $C$ | a time-series $c_1, c_2, \ldots, c_j, \ldots, c_n$ |
| $\boldsymbol{C}$ | a $n\times n$ matrix containing every rotation of $C$ |
| $C_i$ | the $i$th row of the above |
| $Q$ | another time-series $q_1, q_2, \ldots, q_i, \ldots, q_n$ |
| $\bar{Q}$ | a database containing many time-series $=\{Q_1, \ldots, Q_m\}$ |

## 4. WEDGE BASED ROTATION MATCHING

We will begin by showing how we can efficiently search for the best match in the main memory. We will further show how to generalize to other distance measures.

### *4.1. Fast and exact main memory search*

We begin by defining time-series *wedges*. Imagine that we take several time-series, $C_1, \ldots, C_k$, from our matrix $\boldsymbol{C}$. We can use these sequences to form two new sequences $U$ and $L$,

$$U_i = \max(C_{1i}, \ldots, C_{ki}),$$
$$L_i = \min(C_{1i}, \ldots, C_{ki}).$$

$U$ and $L$ stand for upper and lower, respectively. We can see why in figure 4. They form the smallest possible bounding envelope that encloses all members of the set $C_1, \ldots, C_k$ from above and below. More formally,

$$\forall_i \quad U_i \ge C_{1i}, \ldots, C_{ki} \ge L_i.$$

For notational convenience, we will call the combination of $U$ and $L$ a *wedge*, and denote a wedge as $W$,

$$W = \{U, L\}.$$

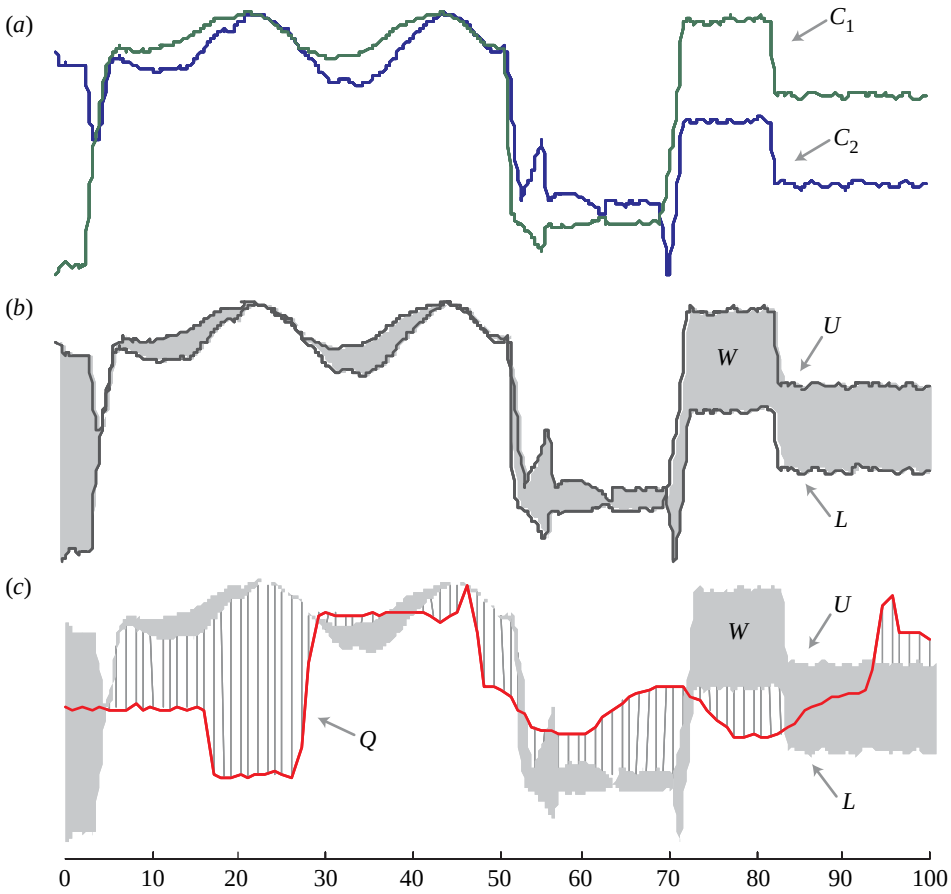


Figure 4. (*a*) Two time-series $C_1$ and $C_2$. (*b*) A time-series wedge $W$, created from $C_1$ and $C_2$. (*c*) An illustration of LB_Envelope.

We can now define a lower-bounding measure between an arbitrary time-series $Q$ and the entire set of candidate sequences contained in a wedge $W$,

$$\text{LB\_Envelope}(Q, W) = \sqrt{\sum_{i=1}^{n} \begin{cases} (q_i - U_i)^2 & \text{if } q_i > U_i \\ (q_i - L_i)^2 & \text{if } q_i < L_i \\ 0 & \text{otherwise} \end{cases}}.$$

For brevity, we do not show a proof of this lower-bounding property. A proof appears in Keogh (2006) and also in Li *et al.* (2004), where the authors use this representation for different problems.

Note that the LB_Envelope function has been used before to support DTW (Keogh 2002; Rath & Manmatha 2002; Vlachos *et al.* 2003; Ratanamahatana & Keogh 2005), uniform scaling (Keogh *et al.* 2004) and query filtering (Wei *et al.* 2005). For these tasks, the lower-bounding distance function is the same, but the definition of $U$ and $L$ is different.

There are two important observations about LB_Envelope. First, in the special case where $W$ is created from a single candidate sequence, it degenerates to the Euclidean distance. Second, not only does LB_Envelope lower bound all the candidate sequences $C_1, \ldots, C_k$, but we can also do *early abandon* with LB_Envelope. While the latter fact might be obvious, for clarity we make it explicit in table 5.

Table 5. LB_Envelope optimized with early abandonment.

```
algorithm [dist, num_steps]=EA_LB_Envelope(Q, W, r)
accumulator=0
for i=1 to length(Q)                    //loop over time-series
 if q_i > W·U_i                         //accumulate error
                                          contribution
     accumulator += (c_i − W·U_i)^2
 elseif q_i < W·L_i
     accumulator += (c_i − W·L_i)^2
 end
 if accumulator > r^2                   //can we abandon?
     return [infinity, i]               //terminate and return
                                          an infinite error
 end                                    //to signal the early
                                          abandonment.
end
return [sqrt(accumulator), length(Q)]   //terminate with true
                                          dist
```

Note once again that the value returned in num_steps is merely a bookkeeping device to allow a post-mortem evaluation of efficiency.

Suppose we have just two time-series, $C_1$ and $C_2$ of length $n$, and we know that in the future we will be given a time-series query $Q$ and asked if one (or both) of $C_1$ and $C_2$ are within $r$ of the query. We naturally wish to minimize the number of steps which we must perform ('steps' are measured by num_steps). We are now in a position to outline two possible approaches to this problem.

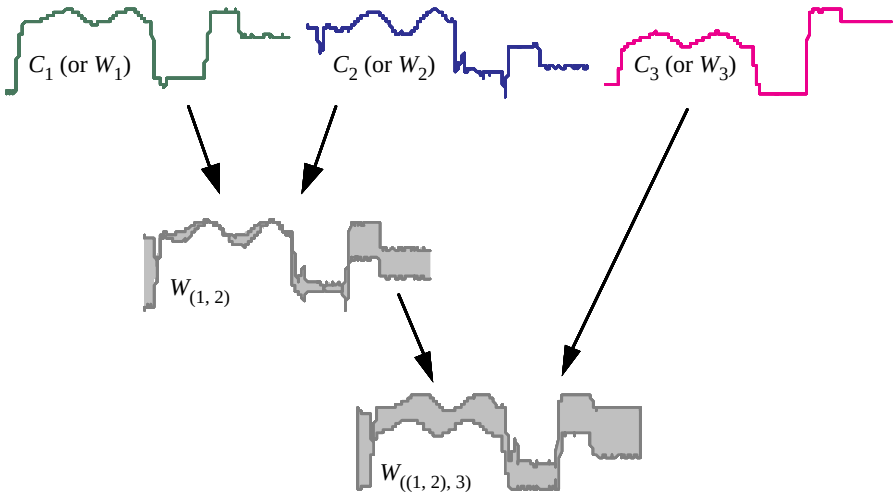


Figure 5. An illustration of hierarchically nested wedges.

— We can simply compare the two sequences, $C_1$ and $C_2$ (in either order), to the query using the early abandon algorithm introduced in table 1. We will call this algorithm, *classic*.
— We can combine the two candidate sequences into a wedge and compare $Q$ to the wedge using LB_Envelope. If the LB_Envelope function early abandons, we are done. We can say with absolute certainty that neither of the two candidate sequences is within $r$ of the query. If we cannot early abandon on the wedge, we need to individually compare the two candidate sequences, $C_1$ and $C_2$ (in either order), to the query. We will call this algorithm, *Merge*.

Let us consider the best and the worst cases for each approach. For *classic*, the worst case is if both candidate sequences are within $r$ of the query, which will require $2n$ steps. In the best case, the first point in the query may be radically different to the first point in either of the candidates, allowing immediate early abandonment and giving a total cost of two steps.

For *Merge*, the worst case is also if both candidate sequences are within $r$ of the query, because we will waste $n$ steps in the lower-bounding test between the query and the wedge, and then $n$ steps for each individual candidate, for a total of $3n$. However, the best case, also if the first point in the query is radically different, would allow us to abandon with a total cost of one step.

Which of the two approaches is better depends on as follows:

— the shapes of $C_1$ and $C_2$. If they are similar, this greatly favours *Merge*.
— the shape of $Q$. If $Q$ is truly similar to one (or both) of the candidate sequences, this would greatly favour *classic*.
— the matching distance $r$. Here, the effect is non-monotonic and dependent on the two factors above.

We can generalize the notion of wedges by hierarchically nesting them. Let us begin by augmenting the notation of a wedge to include information about the sequences used to form it. For example, if a wedge is built from $C_1$ and $C_2$, we will denote it as $W_{(1,2)}$. Note that a single sequence is a special case of a wedge; for example, the sequence $C_1$ can also be denoted as $W_1$. We can combine $W_{(1,2)}$ and $W_3$ into a single wedge by finding maximum and minimum values for each $i$th location, from *either* wedge. More concretely,

$$U_i = \max(W_{(1,2)i}, W_{3i}),$$
$$L_i = \min(W_{(1,2)i}, W_{3i}),$$
$$W_{((1,2),3)} = \{U, L\}.$$

In figure 5, we illustrate this notation. We call $W_{(1,2)}$ and $W_3$ *children* of wedge $W_{((1,2),3)}$. Since individual sequences are special cases of wedges, we can also call $C_1$ and $C_2$ children of $W_{(1,2)}$.

Given the generalization to hierarchal wedges, we can also now generalize the *Merge* approach. Suppose we have a time-series $Q$ and a wedge $W_{((1,2),3)}$. We can compare the query to the wedge using LB_Envelope. If the LB_Envelope function early abandons, we are done. We know with certainty that none of the three candidate sequences is within $r$ of $Q$. If we cannot early abandon on the wedge, we need to compare the two child wedges, $W_{(1,2)}$ and $W_3$, to the query. Again, if we cannot early abandon on the wedge $W_{(1,2)}$, we need to individually compare the two candidate sequences, $C_1$ and $C_2$ (in either order), to the query. We call this algorithm *H-Merge* (Hierarchal Merge).

The utility of a wedge is strongly correlated to its area. We can get some intuition by visually comparing LB_Envelope($Q$, $W_{(1,2)}$) with LB_Envelope ($Q$, $W_{((1,2),3)}$), as shown in figure 6. Note that the area of $W_{((1,2),3)}$ is much greater than that of $W_{(1,2)}$, and that this reduces the value returned by the lower-bound function, and thus the possibility to early abandon.

For some problems, the *H-Merge* algorithm can give exceptionally poor performance. If the wedge $W_{(1,2)}$, created from $C_1$ and $C_2$, has an exceptionally large area (i.e. $C_1$ and $C_2$ are very dissimilar), it is very unlikely to be able to prune off any steps.

At this point, we can see that the efficiency of *H-Merge* is dependent on the candidate sequences and $Q$ itself. In general, merging similar sequences into a hierarchal wedge is a good idea, but merging dissimilar sequences is a bad idea.

The observations above motivate a final generalization of *H-Merge*. Recall that to achieve rotation invariance, we expanded our time-series $C$ into a matrix

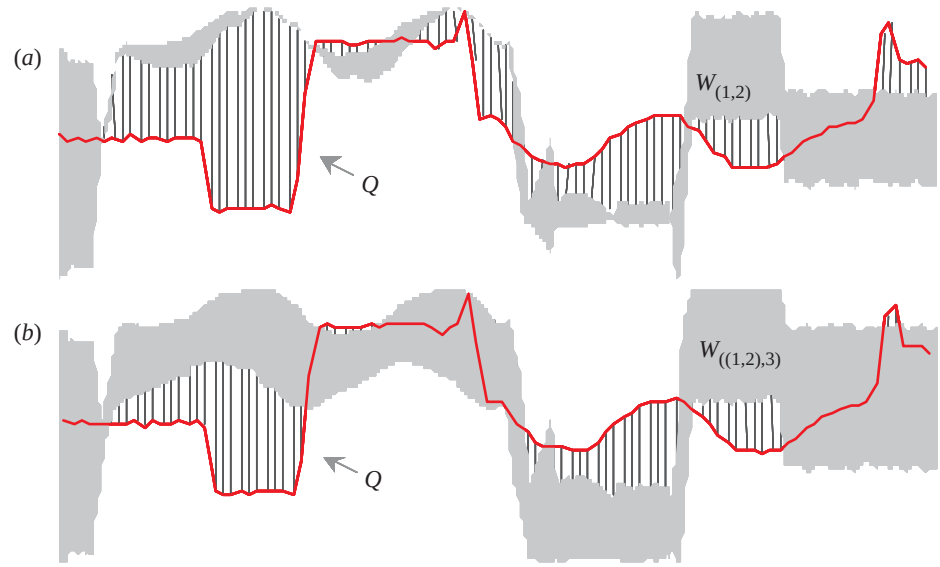


Figure 6. (*a*) An illustration of LB_Envelope($Q$, $W_{(1,2)}$). (*b*) An illustration of LB_Envelope($Q$, $W_{((1,2),3)}$). Note that the tightness of the lower bound is proportion to the number and (squared) length of vertical lines.

Table 6. Algorithm *H-Merge*.

```
algorithm [dist]=H−Merge(Q, W, K, r)
S={empty}                          //initialize a stack.
for i=1 to K                       //place all the wedges
                                     into the stack.
 enqueue(W_set(i),S)
end
while not empty(S)
 T=dequeue(S)
 dist=EA_LB_Envelope               //note that is early
  (Q, T, r)                          abandon version.
 if infinite(dist)                 //we did not early
                                     abandon.
  if cardinality(T)=1              //T was an individual
                                     sequence.
   disp('The sequence ',T, 'is ', dist,
        'units from the query')
   return [dist]
  else                             //T was a wedge, find
                                     its children
   enqueue(children(T),S)          //and push them onto
                                     the stack.
  end
 end
end
```

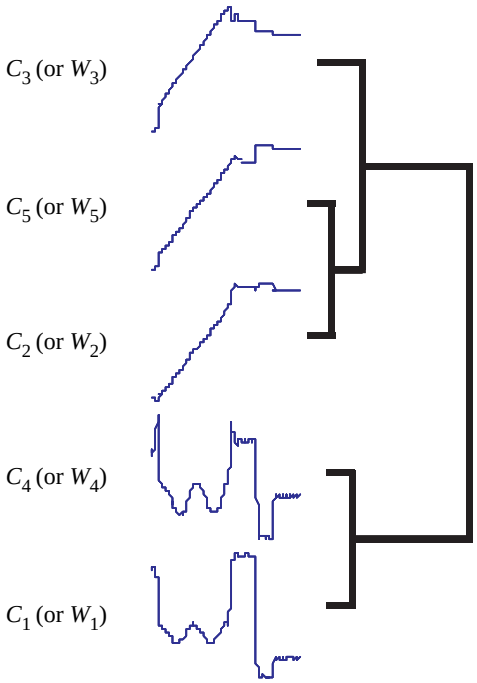


Figure 7. A dendrogram of five sequences $C_1$, $C_2$, …, $C_5$, clustered using group-average linkage.

with $n$ time-series. Given these $n$ sequences, we can merge them into $K$ hierarchal wedges, where $1 \le K \le n$. This merging forms a partitioning of the data, with each sequence belonging to exactly one wedge. We will use $\mathbf{W}$ to denote a set of hierarchal wedges,

$$\mathbf{W} = \{W_{\text{set}(1)}, W_{\text{set}(2)}, \ldots, W_{\text{set}(K)}\}, \quad 1 \le K \le n,$$

where $W_{\text{set}(i)}$ is a (hierarchically nested) subset of the $n$ candidate sequences. Note that we have

$$W_{\text{set}(i)} \cap W_{\text{set}(j)} = \varnothing \quad \text{if } i \neq j, \quad \text{and,}$$
$$|W_{\text{set}(1)} \cup W_{\text{set}(2)} \cup \ldots \cup W_{\text{set}(K)}| = n.$$

We will attempt to merge together only similar sequences. We can then compare this set of wedges against our query. Table 6 formalizes the algorithm.

Note that this algorithm is designed to replace the Test_All_Rotations algorithm that is invoked as a subroutine in the Search_Database_for_Rotated_Match algorithm shown in table 3.

As we shall see in our empirical evaluations, *H-Merge* can produce very impressive speedup if we make judicious choices in the set of hierarchal wedges that make up $W$. However, the number of possible ways to arrange the hierarchal wedges is greater than $K^K$, and the vast majority of these arrangements will be very poor, so specifying a good arrangement of $W$ is critical.

A simple observation alleviates the need to invent a new algorithm to find a good arrangement of $W$. Note that hierarchal clustering algorithms have very similar

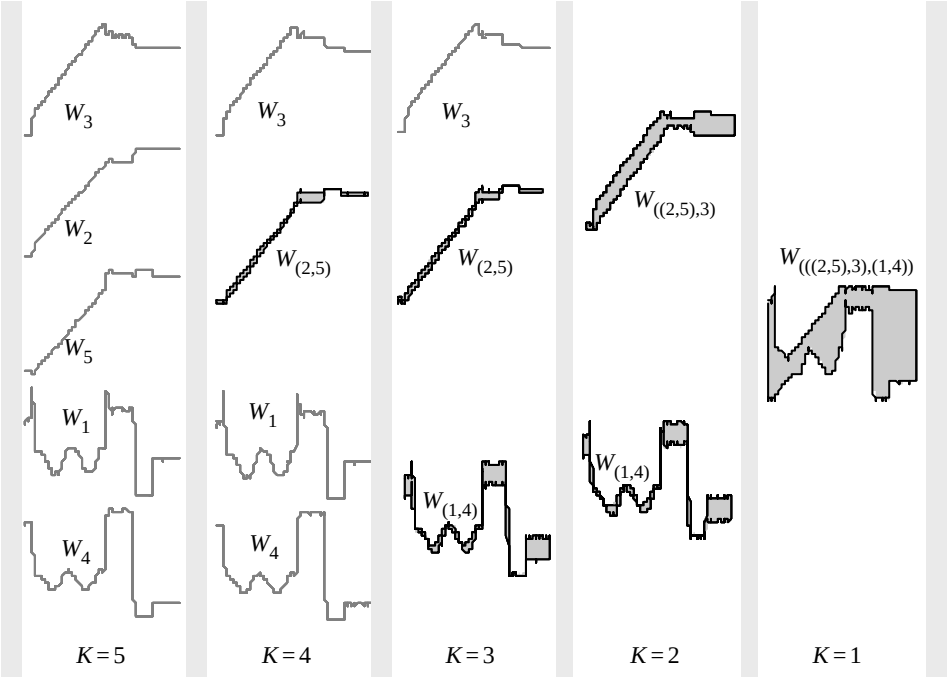


Figure 8. Wedge sets $W$, of size 1–5, derived from the dendrogram shown in figure 7.

goals to an ideal wedge-producing algorithm. In particular, hierarchal clustering algorithms can be seen as attempting to minimize the distances between objects in each subtree. A wedge-producing algorithm should attempt to minimize the area of each wedge. However, the area of a wedge is simply the maximum Euclidean distance between any sequences contained therein (i.e. Newton–Cotes rule from elementary calculus). This motivates us to derive wedge sets based on the result of a hierarchal clustering algorithm. Figure 8 shows wedge sets $W$, of every size from 1 to 5, derived from the dendrogram shown in figure 7.

Given that the clustering algorithm produces the tentative wedge sets, all we need to do is to choose the best one. We could attempt to do this by eye; for example, in figure 8, it is clear that any sequence that early abandons on $W_3$ will almost certainly also early abandon on both $W_2$ and $W_5$; similar remarks apply to $W_1$ and $W_4$. At the other extreme, the wedge at $K=1$ is so 'fat' that it is likely to have poor pruning power. The set $W=\{W_{((2,5),3)}, W_{(1,4)}\}$ is probably the best compromise. However, because the set of time-series might be very large, such visual inspection is not scalable.

The problem is actually even more complex, in that the best value for $K$ also depends on the current value of $r$ (recall $r$ is the best-so-far in nearest neighbour search.). If $r$ is large, then very little early abandoning is possible, and this favours a large value for $K$. In contrast, if $r$ is small, we can do a lot of early abandoning, and we are better off having many sequences in a single wedge, hence we can early abandon all of them with a single calculation. However, note that for nearest neighbour search, the value of $r$ will get smaller as we search through the database.

With this in mind, we dynamically choose the wedge set based on a fast empirical test. We start with the wedge set where $K=2$. Each time the bestSoFar value changes, we test a subset of the possible values of $K$ and choose the most efficient one (as measured by num_steps) as the next $K$ to use. Which subset to test is decided on-the-fly based on the current $K$ value. They are the values which evenly divide the ranges [1, current_K] and [current_K, max_K] into a small number of intervals. All the experiments below use *five* as this small number; however, the results are not sensitive to this choice. For example, using values as low as 3 or as high as 25 did not change the *efficiency* by more than 1% on datasets larger than 128 objects. Note that on average, the bestSoFar value only changes $\log(m)$ times during a linear search, so this slight overhead in adjusting the parameter is not too burdensome; however, we do include this cost in all the experiments in §5.

### 4.2. Generalizing to other distance measures

As we shall see in §5, the Euclidean distance is typically very effective and intuitive as a distance measure for shapes. However, in some domains, it may not produce the best possible precision/recall or classification accuracy (Adamek & O'Connor 2003; Ratanamahatana & Keogh 2005). The problem is that even after best rotation alignment, subjectively similar shapes may produce time-series that are globally similar but contain local 'distortions'. These distortions may correspond to local features that are present in both shapes but in different proportions. For example, in figure 9, we can see the more prominent sagittal and nuchal crests of a lowland gorilla change the locations, in which the brow ridge and jaw map to in a time-series relative to a mountain gorilla. However, it should be noted that the two specimens do not differ much in the actual shape of the brain case, much of the difference attributable to the extracranial structures such as the sagittal and nuchal crests.

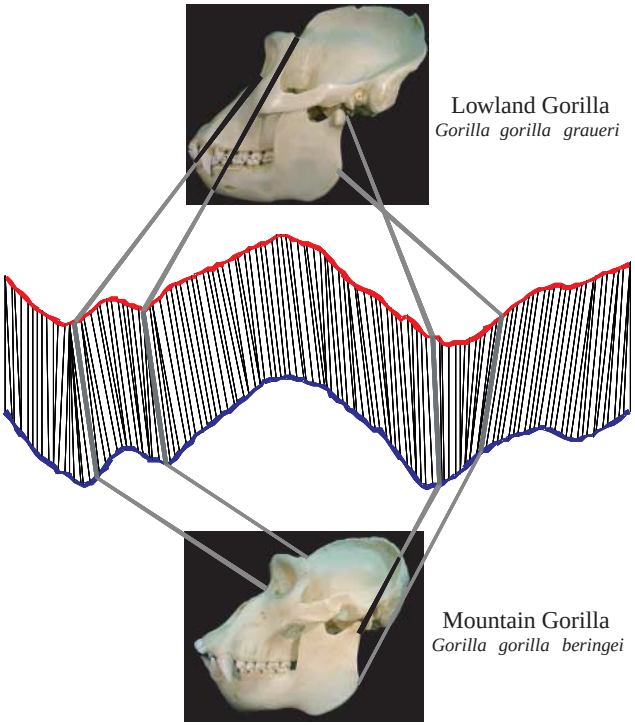


Figure 9. Lowland and mountain gorillas are morphologically similar, but have slightly different proportions. Dynamic time warping can be used to align homologous features in the time-series representation space.

Even if we assume that the database contains the actual object used as a query, it is possible that the two time-series are distorted versions of each other. Here, the distortions may be caused by camera perspective effect, differences in lighting causing shadows which appear to be features, parallax, etc.

Fortunately, there is a well-known technique for compensating such local misalignments, dynamic time warping (DTW; Keogh 2002; Ratanamahatana & Keogh 2005). While DTW was invented in the context of one-dimensional speech signals, others have noted its utility for matching shapes, including face profiles (Bhanu & Zhou 2004), leafs (Ratanamahatana & Keogh 2005), handwriting (Rath & Manmatha 2002) and general shape matching (Adamek & O'Connor 2004).

To align two sequences using DTW, an $n\times n$ matrix is constructed, where the ($i$th, $j$th) element of the matrix is the distance $d(q_i, c_j)$ between the two points $q_i$ and $c_j$ (i.e. $d(q_i, c_j)=(q_i-c_j)^2$). Each matrix element $(i, j)$ corresponds to the alignment between the points $q_i$ and $c_j$, as illustrated in figure 10.

A warping path $P$ is a contiguous set of matrix elements that defines a mapping between $Q$ and $C$. The $t$th element of $P$ is defined as $p_t=(i, j)_t$, so we have

$$P = p_1, p_2, \ldots, p_t, \ldots, p_T \quad n \le T < 2n-1.$$

The warping path that defines the alignment between the two time-series is subject to several constraints. For example, the warping path must start and finish in diagonally opposite corner cells of the matrix; the steps in the warping path are restricted to adjacent cells (including diagonally adjacent cells); the points in the warping path must be monotonically spaced in time. In addition to these constraints, virtually all practitioners using DTW also constrain the warping path in a

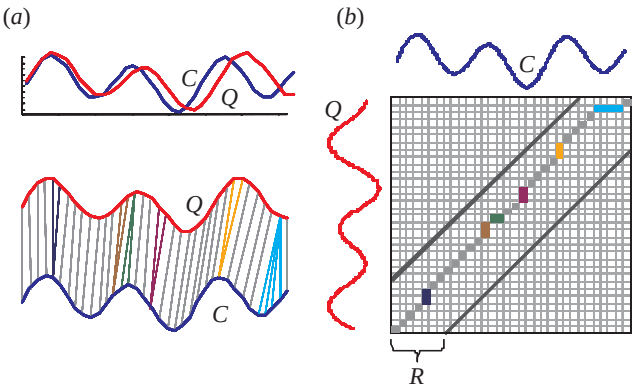


Figure 10. (*a*) Two time-series sequences which are similar but out of phase. (*b*) To align the sequences, we construct a warping matrix and search for the optimal warping path, shown with solid squares. Note that Sakoe–Chiba band with width $R$ is used to constrain the warping path.

global sense by limiting how far it may stray from the diagonal (Keogh 2002; Rath & Manmatha 2002; Ratanamahatana & Keogh 2005). A typical constraint is the Sakoe–Chiba band, which states that the warping path cannot deviate more than $R$ cells from diagonal.

The optimal warping path can be found in $O(nR)$ time by dynamic programming (Keogh 2002). We shall show experimentally in §5 that DTW can significantly outperform Euclidean distance on real datasets.

Based on an arbitrary wedge $W$ and the allowed warping range $R$, we define two new sequences, $DTW\_U$ and $DTW\_L$,

$$\mathrm{DTW\_}U_i = \max(U_{i-R} : U_{i+R}),$$
$$\mathrm{DTW\_}L_i = \min(L_{i-R} : L_{i+R}).$$

They form an additional envelope above and below the wedge, as illustrated in figure 11.

We can now define a lower-bounding measure for DTW distance between an arbitrary query $Q$ and the entire set of candidate sequences contained in a wedge $W$,

$$\mathrm{LB\_Envelope}_{\mathrm{DTW}}(Q, W) = \sqrt{\sum_{i=1}^{n} \begin{cases} (q_i - \mathrm{DTW\_}U_i)^2 & \text{if } q_i > \mathrm{DTW\_}U_i \\ (q_i - \mathrm{DTW\_}L_i)^2 & \text{if } q_i < \mathrm{DTW\_}L_i \\ 0 & \text{otherwise} \end{cases}}.$$

We make the following claim.

**Proposition 4.1.** *For any sequence $Q$ of length $n$ and a wedge $W$ containing a set of time-series $C_1, \ldots, C_k$ of the same length $n$, for any global constraint on the warping path of the form $j-R\le i\le j+R$, the following inequality holds*:

$$\mathrm{LB\_Envelope}_{\mathrm{DTW}}(Q, W) \le \min(\mathrm{DTW}(Q, C_s)), \quad \text{where } s = 1, 2, \ldots, k.$$

Owing to the space limitations, we refer the interested reader to Keogh (2006) for the proof. In addition, space limitations also prohibit a discussion of the minor modifications required to index $\mathrm{LB\_Envelope}_{\mathrm{DTW}}(Q, W)$; however, Vlachos *et al.* (2003) contains the necessary modifications for both DTW and LCSS which are discussed below.

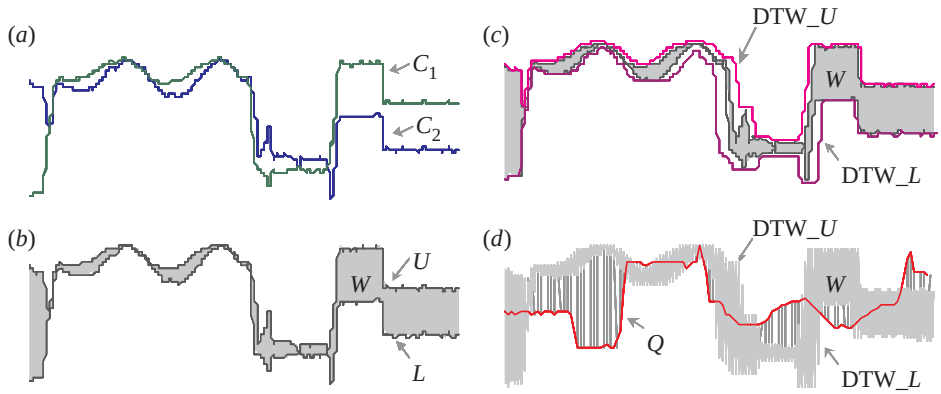


Figure 11. The idea of bounding envelopes introduced in figure 4 is generalized to allow DTW. (*a*) The two time-series $C_1$ and $C_2$. (*b*) A time-series *wedge* $W$, created from $C_1$ and $C_2$. (*c*) In order to allow lower bounding of DTW, an additional envelope is created above and below the wedge. (*d*) An illustration of LB_Envelope$_{\mathrm{DTW}}$.

To facilitate later efficiency comparisons to Euclidean distance and other methods, it will be useful to define the time complexity of DTW in terms of num_steps as returned by tables 1 and 5. The variable 'num_steps' is the number of real-value subtractions that must be performed and completely dominates the CPU time, since the square root function is only performed once (and can be removed; see Keogh & Kasetty 2002). If we construct a full $n \times n$ warping matrix, then DTW clearly requires at least $n^2$ steps. However, as we noted above and illustrated in figure 10, we can truncate the corners of the matrix to reduce this number to approximately $nR$, where $R$ is the width of the Sakoe–Chiba band. While $nR$ is the number of steps for a single DTW, we expect the average number of steps to be less, because some full DTW calculations will not be needed if the lower-bound test fails. Since the lower-bound test requires $n$ steps, the average number of steps when doing $m$ comparisons should be

$$\frac{m \times a(nR) + m(n)}{m},$$

where $a$ is the fraction of the database that requires the full DTW calculated. Note that even this is pessimistic, since both DTW[3] and LB_Envelope$_{\mathrm{DTW}}$ are implemented as early abandoning (recall table 5). Therefore, we simply count the num_steps required by each approach and divide it by $m$ to get the average number of steps required for one comparison.

In addition to DTW, several researchers have suggested using longest common subsequence (LCSS) as a distance measure for shapes (Yazdani & Meral Özsoyoglu 1996). The LCSS is very similar to DTW, except that while DTW insists that every point in $C$ maps onto one (or more) point(s) in $Q$, LCSS allows some points to go unmatched. The intuition behind this idea in a time-series domain is that subsequences may contain additions or deletions; for example, an extra (or forgotten) dance move in a motion capture performance or a missed beat in ECG data. Rather than forcing DTW to produce an unnatural alignment between two

[3]Note that a *recursive* implementation of DTW would always require $nR$ steps; however, *iterative* implementation (as used here) can potentially early abandon with as few as $R$ steps.

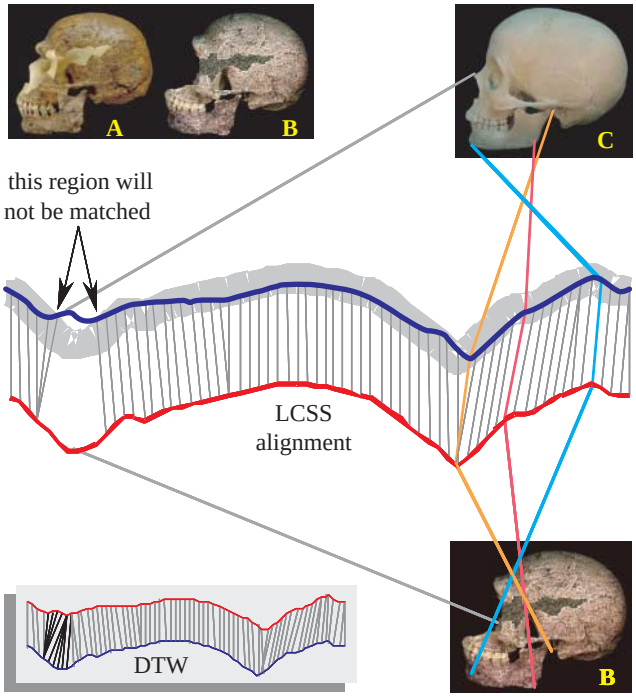


Figure 12. (*a*) Skhul V is usually cast with the missing bones extrapolated in epoxy; however, the original Skhul V (*b*) is missing much of the midface, which means it will match to a modern human (*c*) poorly, even after DTW alignment (inset). In contrast, LCSS alignment will not attempt to match features that are outside a 'matching envelope' (heavy grey line) created from the other sequence.

such sequences, we can use LCSS, which simply ignores parts of the time-series that are too difficult to match. In the image space, the missing section of the time-series may correspond to a partial occlusion of an object or to a physically missing part of the object, as shown in figure 12.

Another example of an anthropological object that requires LCSS to obtain meaningful matches is projectile points as shown in figure 13. Such objects are often discovered with missing parts. In order to find matches using query-by-content in a database, we can try to extrapolate our best guess as to the missing features, but this opens the possibility of imposing our (possibly misguided) knowledge on the problem, rather

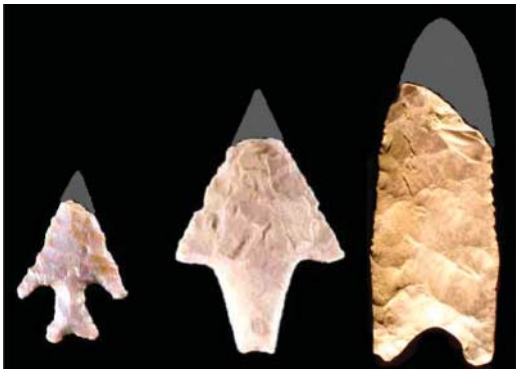

Figure 13. Project points are frequently found with broken tips or tangs. Such objects require LCSS to find meaningful matches to complete specimens. From left to right, Edwards, Langtry and Golondrina projectile points.

Table 7. The error of Euclidean distance and DTW on several publicly available datasets.

| name | number of classes | number of instances | Euclidean error (%) | DTW error (%) $\{R\}$ |
|---|---|---|---|---|
| face | 16 | 2240 | 3.839 | 3.170 {3} |
| Swedish leaves | 15 | 1125 | 13.33 | 10.84 {2} |
| chicken | 5 | 446 | 19.96 | 19.96 {1} |
| mixedBag | 9 | 160 | 4.375 | 4.375 {1} |
| osu leaves | 6 | 442 | 33.71 | 15.61 {2} |
| diatoms | 37 | 781 | 27.53 | 27.53 {1} |

than letting the data speak for itself. Using LCSS, the missing features can be simply ignored during the search process.

While we considered LCSS for generality, we will not further explain how to incorporate it into our framework. It has been shown in Vlachos *et al.* (2003) that it is trivial to lower-bound LCSS using the envelope-based techniques described previously. The minor changes include reversing some inequality signs, since LCSS is a similarity measure, not a distance measure. Our omission here of a detailed discussion is due to space limitations and to a slight bias against the method. Unlike Euclidean distance which has no parameters, or DTW, which has one intuitive and easy-to-set parameter, LCSS requires two parameters, and tuning them is non-trivial. In the experiments, we found that we could sometimes tune LCSS to *slightly* beat DTW on *some* problems; however, we did not have large enough datasets to allow training/test splits that guarded against overfitting to a statistically significant standard.

## 5. EXPERIMENTAL RESULTS

In this section, we empirically evaluate our approach. We begin by stating our experimental philosophy. In a recent paper, Veltkamp & Latecki (2006) attempted to reproduce the accuracy claims of several shape matching papers, but discovered to their dismay that they could not match the claimed accuracy for any approach. One suggested reason is

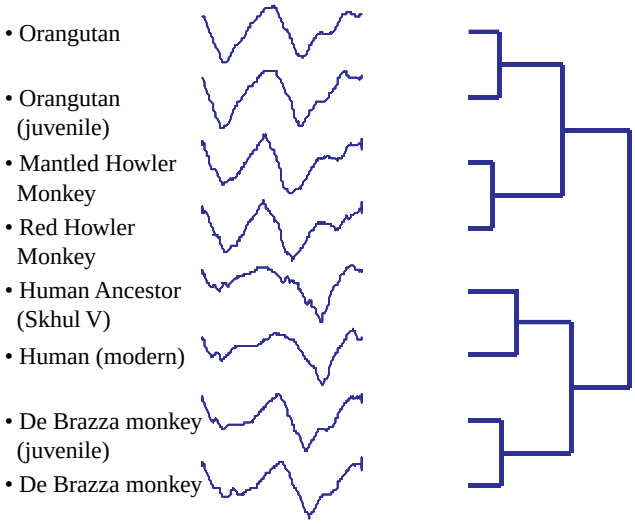


Figure 14. A group-average hierarchal clustering of eight primate skulls based on the lateral view, using Euclidean distance.

the observation that many approaches have highly tuned parameters, a fact which we believe makes Euclidean distance (zero parameters) and DTW (one parameter) particularly attractive. Veltkamp & Latecki conclude that 'it would be good for the scientific community if the reported test results are made reproducible and verifiable by publishing datasets and software along with the articles'. We completely concur and have placed *all* datasets at the following URL (Keogh 2006).

### 5.1. Effectiveness of shape matching

In general, this paper is not making any claims about the *effectiveness* of shape matching. Because we are simply speeding up arbitrary distance calculations on arbitrary one-dimensional representations of shapes, we automatically inherit the well-documented effectiveness of other researchers' published work (Gdalyahu & Weinshall 1999; Adamek & O'Connor 2003, 2004; Attalla & Siy 2005; Jalba *et al.* 2005; Ratanamahatana & Keogh 2005; Vlachos *et al.* 2005).

Nevertheless, for completeness, and in order to justify the extra computational expense of DTW, we will show the effectiveness of shape matching on several publicly available datasets.

Table 7 shows the error rate of one-nearest neighbour classification as measured using leaving-one-out evaluation. Recall that Euclidean distance has no parameters, DTW has a single parameter (the warping window width $R$), which was learned by looking only at the training data. For the face and leaf datasets, the (approximate) correct rotation was known (Ratanamahatana & Keogh 2005). We removed this information by randomly rotating the images.

The mixedBag dataset is small enough to run the more computationally expensive chamfer (Borgefors 1988) and Hausdorff (Olson & Huttenlocher 1997) distance measures. They achieved an error rate of 6.0 and 7.0%, respectively (Vlachos *et al.* 2005), slightly worse than Euclidean distance. Likewise, the chicken dataset allows us to compare directly to Mollineda *et al.* (2002), which

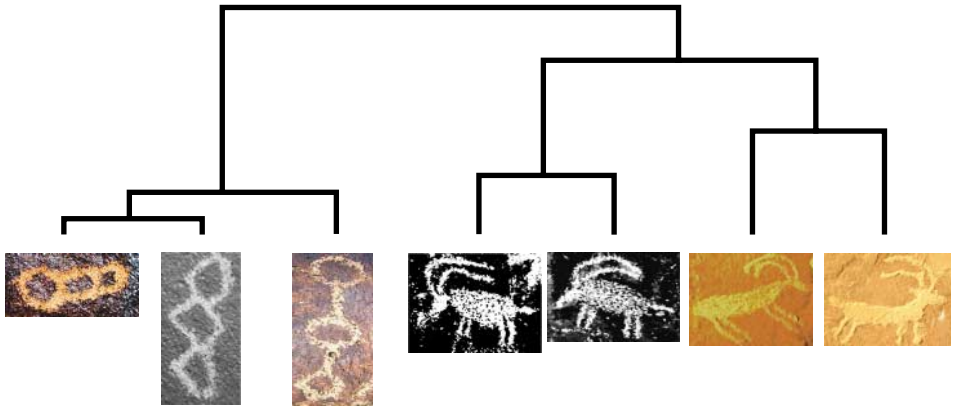

Figure 15. A group-average hierarchal clustering of seven petroglyphs from Nevada and California, using DTW distance.

used identical experiments to test six different algorithms based on *discrete* sequences extracted from the shapes. The best of these algorithms had an error rate of 20.5% and took over a minute for each distance calculation, whereas our approach takes an average time of 0.0039 s for each distance calculation.[4] For the diatom dataset, the results are competitive with human experts, whose error rates ranged from 57 to 13.5% (Jalba *et al.* 2005), and only slightly worse than the morphological curvature scale spaces (MCSS) approach of Jalba *et al.* (2005), which obtained 26.0%. However, note that the Euclidean distance requires zero parameters once the time-series have been extracted, whereas the MCSS has several parameters to set.

In general, these experiments show two things (which had been noted before): the extra effort of DTW is useful in some domains and very simple time-series representations of shapes are competitive to other more complex representations.

We also performed extensive 'sanity check' experiments using a large database of primate skulls. For all species where we have at least two examples, we perform a hierarchical clustering and check to see if both samples of the same species clustered together. Figure 14 shows a typical example.

It is important to recall that figure 14 shows a phenogram, *not* a phylogenetic tree. However, on larger-scale experiments in this domain (shown in Keogh 2006), we found that large subtrees of the dendrograms did conform to the current consensus on primate evolution (Fleagle 1999).

We performed many additional experiments with large collections of petroglyphs and projectile points. Here, the evaluation is more subjective, but we find that in most cases, we get very intuitive results as in figure 15.

In most cases, we discovered that unintuitive results were caused by errors in the image processing software that automatically extracts the shapes, since many petroglyphs are heavily degraded by weather (Pope 2000).

[4]We are aware that one should normally not compare CPU times from different computers; however, here the four orders of magnitude offers a comfortable margin that dwarfs implementation details.

### 5.3. Main memory experiments

There is an increasing awareness that comparing two competing approaches using only CPU time opens the possibility of implementation bias (Keogh & Kasetty 2002). As a simple example, while the Haar wavelet transform is $O(n)$ and DFT is $O(n \log n)$, the DFT is *much* faster in the popular language MATLAB, simply because it is a highly optimized subroutine. For this reason, many recent papers compare approaches with some implementation-free metric (Keogh 2002; Vlachos *et al.* 2003; Ratanamahatana & Keogh 2005; Vlachos *et al.* 2005). As we noted earlier, the variable 'num_steps' returned by tables 1 and 5 allows an implementation-free measure to compare performance.

For Euclidean distance queries, we compare to *brute force* and *fast Fourier transform* (FFT) methods, which are the only competitors to also guarantee no false dismissals (Vlachos *et al.* 2005). The cost model for the FFT lower bound is $n \log n$ steps. If the FFT lower bound fails, we allow the approach to avail of our early abandoning techniques discussed in §3.

We tested on two datasets, a homogeneous database of 16 000 projectile point images, all of length 251 and a heterogeneous dataset consisting of all the data used in the classification experiments, plus 1000 projectile points. In total, the heterogeneous dataset contains 5844 objects of length 1024. To measure the performance, we averaged over 50 runs, with the query object randomly chosen and removed from the dataset.

We measure the average number of steps required by each approach for a single comparison of two shapes, divided by the number of steps require by brute force. For our method, we include a start-up cost of $O(n^2)$, which is the time required to build the wedges. Because the utility of early abandoning depends on the value of the best-so-far, we expect our method to do better as we see larger and larger datasets.

Figure 16 shows the results on the projectile points dataset using Euclidean distance.

We can see that for small datasets, our approach is slightly worse than *FFT* and simple *Early abandon* because we had to spend some time building the wedges. However, by the time we have seen 64 objects, we have already broken even, and thereafter rapidly race towards beating *FFT* and *Early abandon* by one order of magnitude and *Brute force* by two orders of magnitude.

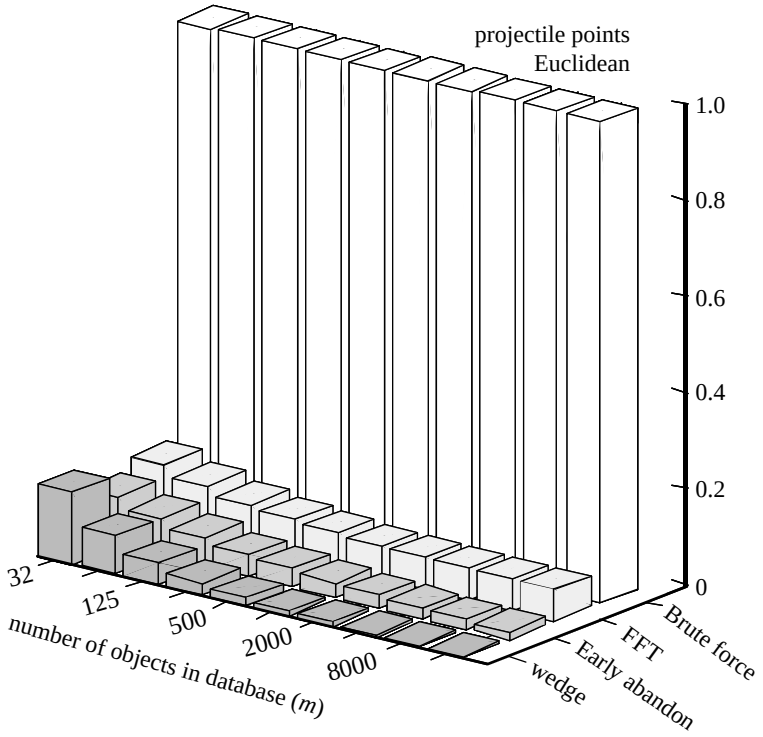


Figure 16. The relative performance of four algorithms on the projectile points dataset using the Euclidean distance measure.

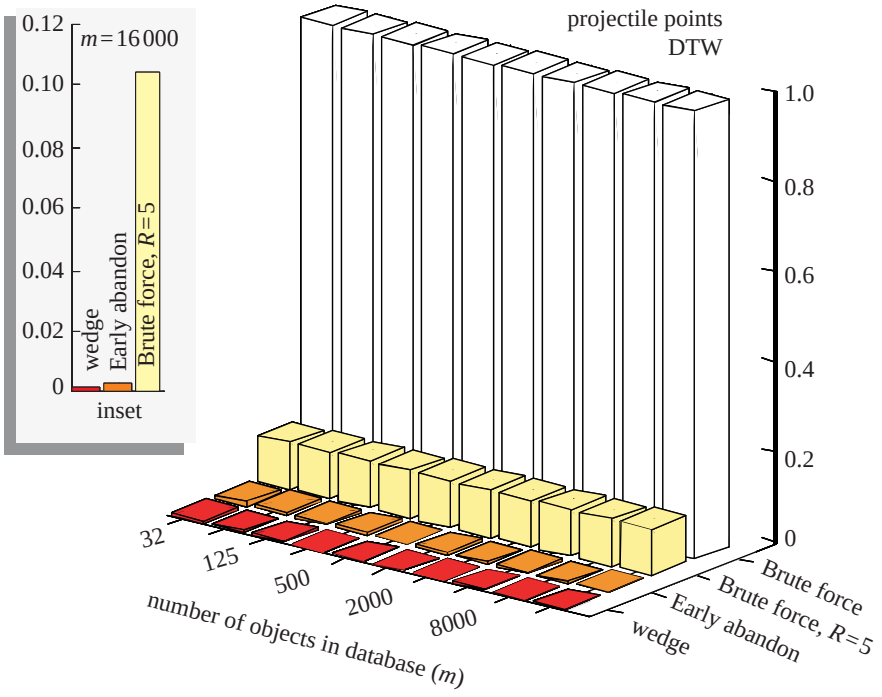


Figure 17. The relative performance of four algorithms on the projectile points dataset using the DTW distance measure. The inset shows a zoom-in of the three best algorithms when $m=16\,000$.

The results on the projectile points dataset using DTW are shown in figure 17, and are even more dramatic.

Here, the cost of building the wedges is dwarfed by a single brute force–DTW–rotation invariant comparison, so our approach is faster even for a database of size 3. By the time we have examined the entire database, our approach is more than 5000 times faster than the brute force approach. It is interesting to note that the early abandoning strategy is by itself quite competitive, yet to our knowledge no one uses it. We suspect that this is because most people are more familiar with the elegant and terse recursive version of DTW, which does not allow early abandoning, than the iterative implementation, which does. However, note that even though our highly optimized early abandoning strategy is competitive, our wedge approach is still an order of magnitude faster once the dataset is larger than 500 objects.

Sometimes indexing methods that work well for highly homogeneous datasets do not work well for heterogeneous

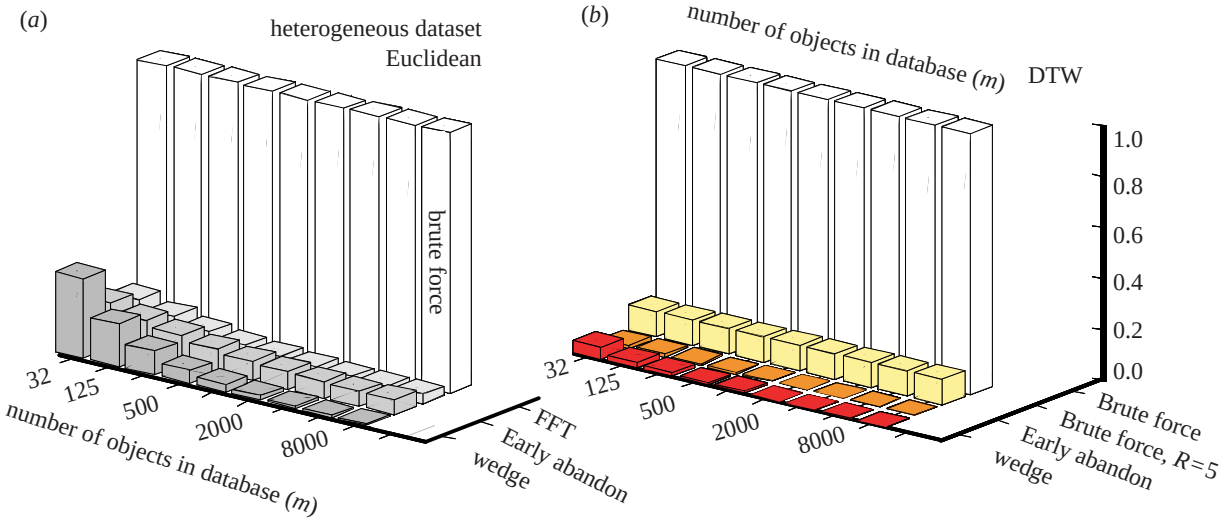


Figure 18. The relative performance of four algorithms on the heterogeneous dataset, using (*a*) Euclidean distance and (*b*) DTW.

datasets, and vice versa. We consider this possibility by testing on the heterogeneous dataset in figure 18.

In this dataset, it takes our wedge approach slightly longer to beat *Early abandon* (and *FFT* for Euclidean search); however, by the time we have seen 8000 objects, our approach is two orders of magnitude faster than its Euclidean competitors, and for DTW it is an order of magnitude faster than *Early abandon* and 3976 times faster than brute force.

Recall that our algorithm requires the setting of a single parameter, the number of intervals to search for a new value for $K$ every time the bestSoFar variable is updated. In all the experiments above, this value was set to 5. We found that we can change this value to any number in the range 3–20 without affecting the performance of our algorithm by more than 4%; therefore, we omit further discussion of this parameter setting.

As a final sanity check, we also measured the wall clock time of our best implementation of all methods. The results are essentially identical to those shown previously.

## 6. CONCLUSIONS AND FUTURE WORK

We have introduced a method to support fast rotation invariant search of large shape datasets with arbitrary representations and distance functions. Our method supports rotation-limited queries and mirror image invariance if desired. This contrasts with most rotation invariant feature-based approaches, which are permanently unable to distinguish between rotations or mirroring. We have shown that our method gives impressive speedup for the state-of-the-art methods without sacrificing the guarantee of no false dismissals.

Note that our work assumes the availability of high quality and high contrast images. In some domains, especially petroglyphs, such images are difficult to obtain (Pope 2000); it would be interesting to consider methods that are more robust to error introduced in this way.

Future work includes both extensions and applications of the current work. We will attempt to extend this approach to the indexing of three-dimensional shapes, and we have begun to use our algorithm as a subroutine in several data mining algorithms, which attempt to cluster, classify and discover motifs in a variety of anthropological datasets, including petroglyph and projectile point databases.

We gratefully acknowledge many useful comments from the anonymous reviewers, as well as helpful comments from Chotirat Ann Ratanamahatana, Michail Vlachos and Longin Jan Latecki. Thanks to Jason Dorff for help with skull images.

## APPENDIX A: A NOTE ON NOTATION

The text of this paper features 'Big $O$' notation, such as '…DTW is $O(n^2)$'. Big $O$ is a mathematical notation used to describe the asymptotic behaviour of functions. Its purpose is to characterize a function's behaviour for large inputs in a simple but rigorous way that enables comparison to other functions. More precisely, it is used to describe an asymptotic upper bound for the magnitude of a function in terms of another, usually simpler, function. It is useful in the analysis of the complexity of algorithms. For example, suppose we have a list of $n$ objects, we want the computer to sort, perhaps we have 32 primate skulls we want to sort by mass. If we say that a particular sorting algorithm is $O(n^2)$, we mean that this algorithm will take approximately $n^2$ steps to do the sorting, in this case 1024 steps. In contrast, a better sorting algorithm might be described as being $O(n \log_2(n))$, which means that it would take approximately $32 \log_2(32) = 160$ steps. Note that the difference between these two algorithms is not fixed, but get worse for larger $n$. See Knuth (1997) for a more detailed treatment.

## REFERENCES

Adamek, T. & O'Connor, N. E. 2003 Efficient contour-based shape representation and matching. *Multimedia Inf. Ret. 2003*, 138–143.

Adamek, T. & O'Connor, N. E. 2004 A multiscale representation method for nonrigid shapes with a single closed contour. *IEEE Circuits Syst. Video Technol.* **14**, 742–753. (doi:10.1109/TCSVT.2004.826776)

Agrawal, R., Faloutsos, C. & Swami, A. N. 1993 Efficient similarity search in sequence databases. In *Proc. 4th Int. Conf. Found. Data Organ. Algorithms (FODO)*, pp. 69–84. Chicago, IL: Springer.

Attalla, E. & Siy, P. 2005 Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching. *Pattern Recognit.* **38**, 2229–2241. (doi:10.1016/j.patcog.2005.02.009)

Bhanu, B. & Zhou, X. 2004 Face recognition from face profile using dynamic time warping. In *Proc. Int. Conf. Pattern Recognit. (ICPN'04)*, pp. 499–502.

Borgefors, G. 1988 Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.* **10**, 849–865. (doi:10.1109/34.9107)

Cardone, A., Gupta, S. K. & Karnik, M. 2003 A survey of shape similarity assessment algorithms for product design and manufacturing applications. *ASME J. Comput. Inf. Sci. Eng.* **3**, 109–118. (doi:10.1115/1.1577356)

Fleagle, J. G. 1999 *Primate adaptation and evolution*. San Diego, CA: Academic Press.

Gdalyahu, Y. & Weinshall, D. 1999 Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**, 1312–1328. (doi:10.1109/34.817410)

Jalba, A. C., Wilkinson, M. H. F., Roerdink, J. B. T. M., Bayer, M. M. & Juggins, S. 2005 Automatic diatom identification using contour analysis by morphological curvature scale spaces. *Mach. Vis. Appl.* **16**, 217–228. (doi:10.1007/s00138-005-0175-8)

Karydis, Y., Nanopoulos, A., Papadopoulos, A. N. & Manolopoulos, Y. 2005 Evaluation of similarity searching methods for music data in peer-to-peer networks. *Int. J. Bus. Intell. Data Mining* **1**, 210–228. (doi:10.1504/IJBIDM.2005.008363)

Keogh, E. 2002 Exact indexing of dynamic time warping. In *Proc. 28th Int. Conf. Very Large Data Bases*, pp. 406–417.

Keogh, E. 2006 www.cs.ucr.edu/~eamonn/shape/shape.htm.

Keogh, E. & Kasetty, S. 2002 On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*. pp. 102–111.

Keogh, E., Palpanas, T., Zordan, V., Gunopulos, D. & Cardle, M. 2004 Indexing large human-motion databases. In *Proc. 30th Int. Conf. Very Large Data Bases*, pp. 780–791.

Knuth, D. 1997 *The art of computer programming Fundamental algorithms*, vol. 1, 3rd edn. Boston, MA: Addison-Wesley. Section 1.2.11: asymptotic representations, pp. 107–123.

Li, D. & Simske, S. 2002 *Shape retrieval based on distance ratio distribution*. HP Tech Report, HPL-2002-251.

Li, Q., Lopez, I. & Moon, B. 2004 Skyline index for time series data. *IEEE Trans. Knowl. Data Eng.* **16**, 669–684.

Mollineda, R. A., Vidal, E. & Casacuberta, F. 2002 Cyclic sequence alignments: approximate versus optimal techniques. *Int. J. Pattern Recognit. Artif. Intell. (IJPRAI)* **16**, 291–299. (doi:10.1142/S0218001402001678)

O'Brien, M. J. & Lyman, R. L. 2003 Resolving phylogeny: evolutionary archaeology's fundamental issue. In *Essential tensions in archaeological method and theory* (eds T. L. VanPool & C. S. VanPool), pp. 115–135. Salt Lake City, UT: University of Utah Press.

Olson, C. F. & Huttenlocher, D. P. 1997 Automatic target recognition by matching oriented edge pixels. *IEEE Trans. Image Proc.* **6**, 103–113. (doi:10.1109/83.552100)

Osada, R., Funkhouser, T., Chazelle, B. & Dobkin, D. 2002 Shape distributions. *ACM Trans. Graphics* **21**, 807–832. (doi:10.1145/571647.571648)

Pope, G. A. 2000 Weathering of petroglyphs: direct assessment and implications for dating methods. *Antiquity* **74**, 833–843.

Ratanamahatana, C. A. & Keogh, E. 2005 Three myths about dynamic time warping. In *Proc. SIAM Int. Conf. on Data Mining (SDM '05)*, pp. 506–510.

Rath, T. & Manmatha, R. 2002 *Lower-bounding of dynamic time warping distances for multivariate time series*. Tech Report MM-40, University of Massachusetts Amherst.

Veltkamp, R. C. & Latecki, L. J. To appear. Properties and performance of shape similarity measures. In *Proc. IFCS 2006 Conf.: Data Sci. Classification*.

Vlachos, M., Hadjieleftheriou, M., Gunopulos, D. & Keogh, E. 2003 Indexing multi-dimensional time-series with support for multiple distance measures. In *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, pp. 216–225.

Vlachos, M., Vagena, Z., Yu, P. S. & Athitsos, V. 2005 Rotation invariant indexing of shapes and line drawings. In *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, pp. 131–138.

Wang, Z., Chi, Z., Feng, D. & Wang, Q. 2005 Leaf image retrieval with shape features. In *Proc. 4th Int. Conf. Adv. Vis. Inf. Syst.*, pp. 477–487.

Wei, L., Keogh, E., Van Herle, H. & Mafra-Neto, A. 2005 Atomic wedgie: efficient query filtering for streaming time series. In *Proc. 5th IEEE Int. Conf. Data Mining (ICDM 2005)*, pp. 490–497.

White, T. D. 2000 *Human osteology*, 2nd edn. San Diego, CA: Academic Press.

Yazdani, N. & Meral Özsoyoglu, Z. 1996 Sequence matching of images. In *Proc. 8th Int. Conf. Sci. Stat. Database Manag.* pp. 53–62.

Zhang, D. & Lu, G. 2004 Review of shape representation and description techniques. *Pattern Recognit.* **37**, 1–19. (doi:10.1016/j.patcog.2003.07.008)

Zunic, J., Rosin, p. & Kopanja, L. 2006 Shape orientability. In *Proc. 7th Asian Conf. Comput. Vis.*, pp. 11–20.