Fast Best-Match Shape Searching in Rotation Invariant Metric Spaces

Paper 1568988763

ABSTRACT

The matching of two-dimensional shapes is an important problem with applications in domains as diverse as biometrics, industry, medicine and zoology. The distance measure used must be 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 discrimination ability, or in the distance measure, at the expense of efficiency. In this work we explore the metric properties of the rotation invariant distance measures and propose an algorithm for fast similarity searching in the shape space. The algorithm is demonstrated to introduce a dramatic speed-up over the current approaches, and is guaranteed to introduce no false dismissals. The technique avoids a large percentage of the comparisons between a query and the elements at hand, which makes it especially attractive when large shape collections are available and indexing is required.

Keywords

Shape, Rotation Invariance, Triangle Inequality, Burkhard-Keller Search

1. INTRODUCTION

The matching of two-dimensional shapes is an important problem with applications in domains as diverse as biometrics, industry, medicine and zoology. The actual matching process involves two distinct, yet mutually dependant steps. Firstly, a *suitable* representation is selected by mapping the shapes to elements of a certain space (see Figure 1). And secondly, a *suitable* distance measure is defined over the elements of that space. Here, what is meant by *suitable*, is usually a combination that is invariant to scale, shift or rotation transformations and is also robust in the presence of noise. Among these, rotation invariance seems to be uniquely difficult to handle. The fact is recognized by many works in the area such as [14], which notes "*Rotation is always something hard to handle, compared with translation and scaling.*"

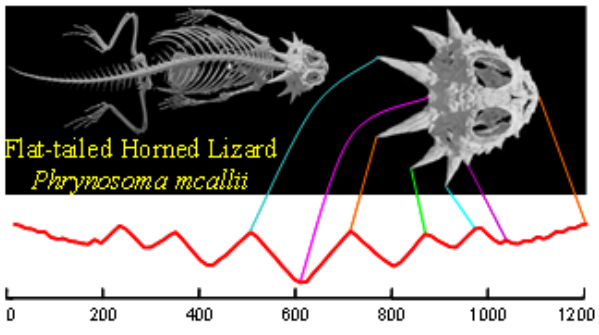

Figure 1: Shapes can be converted to time series. The distance from every point on the profile to the center is measured and treated as the Y-axis of a time series of length n

Many current approaches try to achieve rotation invariance in the representation of the data, at the expense of discrimination ability [17], or in the distance measure, at the expense of efficiency [1, 2, 3, 9]. As an example of the former, the very efficient rotation invariant technique of [17] cannot differentiate between the shapes of the lowercase letters "**d**" and "**b**". As an example of the latter, the work of Adamek et al.[1], which is state of the art in terms of accuracy or precision/recall, takes an untenable $O(n^3)$ for each shape comparison.

In this work we show that one can use some of the more accurate representations, e.g. construct a feature vector (time series) from all shape boundary points as in Figure 1, and still construct a highly efficient matching algorithm. A key point of the proposed approach is that provided certain (reasonable) conditions are met, a rotation invariant distance between the feature vectors defines a metric over the feature space. This observation suggests that a simple, yet highly efficient pruning criterion is applicable. Namely, the triangle inequality. In particular, the proposed technique has the following advantages:

1. The described scheme is applicable for many of the more popular shape representations, e.g. the time series representations described in [1, 3, 7, 21, 22, 24], including the domain specific approaches of [4] and [18].
2. The rotation invariant distance metric can utilize many of the more popular measures for shapes in the literature as the L_p-norms with the Euclidean distance in particular. It is also applicable for non-metric distances as the Dynamic Time Warping (DTW) [2, 4], provided that there exists a metric that lower bounds those measures. For example, the LB_Kim lower-bound [13] has been demonstrated to be such a lower bounding metric for the DTW.
3. As many of the distance computations between a query

and the elements of a dataset are avoided by the proposed algorithm, it is especially attractive when large data collections are considered or disk accesses and indexing are required.

4. In some domains it may be useful to express rotation-limited queries. For example, in order to robustly retrieve examples of the number "**8**", without retrieving the infinity symbol "∞", we can issue a query such as: "Find the best match to this shape allowing a maximum rotation of 15 degrees". Our algorithm supports such rotation-limited queries.

The rest of this paper is organized as follows. In Section 2 we discuss background material and related work. In Section 3 we formally introduce the problem and in Section 4 we offer our solution. Section 5 presents a comprehensive empirical evaluation of the proposed algorithm. Finally, Section 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 [7, 20, 24] for excellent surveys. While not all work on shape matching uses a 1D representation of the 2D shapes, an increasingly large majority of the literature does. We therefore only consider 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 [5] and Hausdorff [16] distance measures, require $O(n^2 \log n)$[1] time and recent experiments (including some in this work) suggest that 1D 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 below.

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 flavors, 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 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 nose [4]. 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 or glasses. Probably the only reason any progress has been made in this area is that most work reasonably assumes that faces presented in an image are likely to be upright.

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 increasing recognition that the "... *major axis is sensitive to noise and unreliable*" [24]. For example, a recent paper shows that under some circumstances, a single extra pixel can change the rotation by 90 degrees [25].

To show how brittle landmarking can be, we performed a simple clustering experiment, where three skulls are clustered using the Euclidean distance with both the major axis technique and the minimum distance of all possible rotations (as found by brute force search). Figure 2 shows the result.

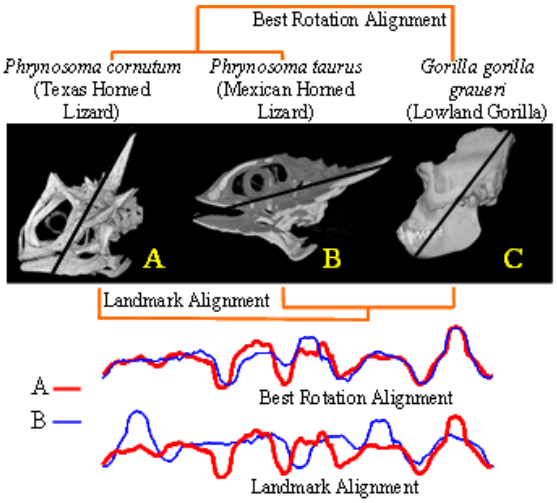

Figure 2: *Top*: Two skulls of lizards from the same genus, and a primate skull are hierarchically clustered using both the landmark rotation beginning at the major axis, and the best rotation. *Bottom*: The landmark-based alignment of **A** and **B** demonstrates why the landmark-based clustering is unsuitable: a small amount of rotation error results in a large difference in the distance measure.

The most important lesson we learned from this experiment (and dozens of other similar experiments on diverse domains) is that rotation (mis)alignment is the most important invariance for shape matching. Unless we have the best rotation, the results will often be inaccurate.

2.2 Rotation Invariant Features

A large number of papers achieve fast rotation invariant matching by extracting only rotation invariant features and indexing them with a feature vector [7]. 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 [17] 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, as it can trivially also handle 3D 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 [7]. In retrospect this is hardly

[1]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. If accurate representation and no false dismissals are required, both R and p should be set to n.

surprising. In order to achieve rotation invariance, all rotation information must be discarded; inevitably, some useful information will also be discarded in this process too. Our experience with these methods suggests that they can be useful for making quick coarse discriminations, for example differentiating between skulls and vertebrae. However, we could not get these methods to distinguish between the skulls of some reptiles (e.g. crocodiles and turtles, see Figure 3), a trivial problem for a human or the brute force algorithm discussed in the next section.

2.3 Brute Force Rotation Alignment

Many authors recognize that the above attempts at approximating rotation invariance are unsatisfactory for most domains, and they achieve true rotation invariance by exhaustive brute force search over all possible rotations, but only at the expense of computational efficiency and indexability [1, 2, 3, 9, 22]. For example, paper [1] uses DTW to handle nonrigid shapes in the time series domain. While they note that most invariances are trivial to handle in this representation, they state "*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 $O(n^3)$ and forces them to abandon hope of indexing. Similarly [22] notes "*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.*" All the techniques introduced thus far to mitigate this untenable computational complexity do so at the expense of introducing false dismissals. Typically they offer some implicit or explicit trick to find a one (or a small number of) of starting point(s) [2, 3, 9]. For example [2] suggests "*In order to avoid evaluation of the dissimilarity measure for every possible pair of starting contour points we propose to extract a small set of the most likely starting points for each shape.*" Furthermore, both the heuristic used and the number of starting points must "*be adjusted to a given application*", and it is not obvious how to best achieve this. In forceful experiments on publicly available datasets it has been demonstrated that brute force rotation alignment produces the best precision/recall and accuracy in diverse domains [1, 2]. In retrospect this is not too surprising. The rival techniques with rotation invariant features are all using some lossy transformation of the data. In contrast the brute force rotation alignment techniques are using a (potentially) lossless transformation of the data. With more high quality information to use, any distance measures will have an easer time reflecting the true similarity of the original images. The contribution of this work is to speed up these accurate but slow methods by many orders of magnitude while producing identical results.

3. ROTATION INVARIANT MATCHING

We begin by formally defining the rotation invariant matching problem. For clarity of presentation we will generally refer to "time series", which the reader will note can be mapped back to the original shapes.

Let $\Omega = \{C_i\}$ be the space of all time series of length n (i.e. $C_i = (c_1, c_2, \ldots, c_n)$), extracted from shapes with a particular method. The shape matching problem searches for the most *similar* element to a given query $Q \in \Omega$ within a subset $\widehat{\Omega} \subset \Omega$ of m time series (i.e. $\widehat{\Omega} = \{C_1, C_2, \ldots, C_m\}$). As we are interested in large data collections, usually we have $m \gg n$.

The similarity between Q and an arbitrary time series $C_i \in \Omega$ is measured in terms of a preselected distance function $D(Q, C_i)$ (e.g. the Euclidean distance), defined over the entire space Ω. If the series are aligned correctly, then the distance function, if suitable in general, will usually provide a good measure of similarity. However, if the shapes are not rotation aligned, then the corresponding time series will be misaligned too and the distance measure might produce extremely poor results. To overcome this problem we need to hold one shape fixed, rotate the other, and record the minimum distance to all possible rotations.

In terms of the time series representation, the above rotations of the series C_i yield a *rotation matrix* $\mathbf{C}_i$ of size n by n:

$$\mathbf{C}_i = \begin{pmatrix} c_1 & c_2 & \ldots & c_{n-1} & c_n \\ c_2 & \ldots & c_{n-1} & c_n & c_1 \\ & & \vdots & & \\ c_n & c_1 & c_2 & \ldots & c_{n-1} \end{pmatrix}$$

It will be useful below to address the time series in each row individually, so we will further denote the j-th row of $\mathbf{C}_i$ as C_i^j.

The *Rotation invariant Distance* (RD) can now be defined as:

$$RD(Q, C_i) = \min_{1 \le j \le n} D(Q, C_i^j) \quad (1)$$

The time complexity for computing the most similar shape to the query using the above rotational distance is $O(mnk)$, where $O(k)$ is the complexity of computing the distance function D. For example, if D is any of the L_p-norms, the complexity of the nearest neighbor search using RD as distance measure becomes $O(mn^2)$. When online processing of large number of queries is required or when the dataset is very large, this running time is simply untenable.

In the next section we propose a procedure that can reduce the running time significantly by simply using the pruning capability of the triangle inequality. Before we continue, we review the notation introduced thus far (see Table 1)

Table 1: Notation Table

C_i	A *shape* time series $C_i = (c_1, c_2, \ldots, c_n)$
Ω	The space of all C_i
$\mathbf{C}_i$	Matrix containing every rotation of C_i
C_i^j	The j-th row of the above rotation matrix
$\widehat{\Omega}$	Dataset of time series $\widehat{\Omega} \subset \Omega$
Q	A query time series $Q = (q_1, q_2, \ldots, q_n) \in \Omega$
$D(Q, C_i)$	*Inner* distance function between Q and C_i
$RD(Q, C_i)$	Rotation distance between Q and C_i

There are two simple and useful generalizations of these definitions:

Mirror Image Invariance: Depending on the application, we may wish to retrieve shapes that are *enantiomorphic* (mirror images) to the query. For example, in matching skull profiles, 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 $\mathbf{C}_i$ to contain the rotations of both C_i and its reverse.

Rotation-Limited Invariance: In some domains it may be useful to express *rotation-limited* queries. For example, in

order to robustly retrieve examples of the number "6", without retrieving examples of the number "9", we can issue a query such as: "Find the best match to this shape allowing a maximum rotation of ± 15 degrees". Our framework trivially supports such rotation-limited queries, by ignoring from consideration those time series from the matrix $\mathbf{C}_i$ that correspond to the unwanted rotations.

4. BEST-MATCH SHAPE SEARCHING

As pointed out, searching for the most similar shape to a given query in the dataset $\widehat{\Omega}$ can easily become intractable as its size increases. We demonstrate a simple property of the rotation invariant distance that allows one to perform highly efficient best-match searches, regardless of the size of the dataset. Namely, that the rotation invariant distance $RD(Q, C_i)$ defines a *pseudo-metric* over the space Ω.

4.1 Rotation Invariant Metric Spaces

The distance function $D(C_i, C_j)$ is said to be a *metric* over the space Ω if for arbitrary elements $C_i, C_j, C_k \in \Omega$ it satisfies the following three properties:

1. *Positivity*: $D(C_i, C_j) \geq 0$, with equality iff $C_i = C_j$
2. *Symmetry*: $D(C_i, C_j) = D(C_j, C_i)$
3. *Triangle inequality*: $D(C_i, C_j) + D(C_k, C_j) \geq D(C_i, C_k)$

When only the second and the third of the above properties are satisfied, $D(C_i, C_j)$ is said to define a pseudo-metric. Showing that a distance function satisfies the triangle inequality is of particular importance when working with large datasets, as it can significantly decrease the searching time by excluding from consideration many of the dataset elements. A number of techniques that utilize the triangle inequality have been proposed over the years, e.g. [6, 8], as well as some popular indexing structures as the Vantage Point trees [23]. Here we show that, provided the inner distance satisfies the triangle inequality, the rotation distance satisfies it too.

PROPOSITION 1. *If the inner distance $D(C_i, C_j)$ is a pseudo-metric over the space of the shape time series Ω, then the rotation invariant distance $RD(C_i, C_j)$ also defines a pseudo-metric over Ω.*

PROOF. Without loss of generality, assume that C_i^j is the rotation of C_i that has a minimal inner distance to C_j, i.e. $RD(C_i, C_j) = D(C_i^j, C_j)$. Similarly we have $RD(C_k, C_j) = D(C_k^j, C_j)$ and $RD(C_i, C_k) = D(C_i^k, C_k)$.
Symmetry: The following equalities hold:

$$RD(C_i, C_j) = D(C_i^j, C_j) = D(C_j, C_i^j) = RD(C_j, C_i)$$

In the above we used the fact that D is symmetric and that the best alignment of the two series is the same regardless of which one of the two we rotate.
Triangle inequality: The following holds:

$$\begin{aligned} RD(C_i, C_j) + RD(C_k, C_j) &= D(C_i^j, C_j) + D(C_k^j, C_j) \\ &\geq D(C_i^j, C_k^j) \geq D(C_i^k, C_k) \\ &= RD(C_i, C_k) \end{aligned}$$

The first inequality above is true as D satisfies the triangle inequality, and the second one follows from the fact that $D(C_i^k, C_k)$ is the distance between the optimal alignment of C_i and C_k, while (C_i^j, C_k^j) also corresponds to some possible alignment. □

The symmetry property is important as it allows us to perform only unidirectional computation of the distances between the series. If D is a metric rather than pseudo-metric and we assume that out of all rotated versions C_i^j of a time series C_i only one is present in the dataset, then RD satisfies the positivity property too, so it is also a metric. The lack of the property however, does not influence the pruning ability of the distance measure introduced by its symmetry and the triangle inequality.

Requiring D to satisfy the triangle inequality may seem restrictive for the rotation distance. However, some of the distance functions that have been demonstrated to perform best for time series analysis are metrics, e.g. the L_p-norms with the Euclidean distance (L_2) in particular (see Section 5.1). Even if the distance function D does not satisfy the triangle inequality, it can still be used as a pruning criterion provided that there exists a lower bounding function LB_D (i.e. $LB\_D(C_i, C_j) \leq D(C_i, C_j), \forall C_i, C_j \in \Omega$) which is a metric. For example, for one of the popular distance functions, the Dynamic Time Warping (DTW), such a metric bounding function has been demonstrated to be the LB_Kim [13] lower bound. In general, the tighter the lower bounding metric that we find, the better the pruning capability of any algorithm utilizing the triangle inequality.

Next we demonstrate how the obtained result can be used for building an efficient best-match searching algorithm for rotation invariant shapes.

4.2 Efficient Best-Match Searching

This section introduces a scheme for fast rotation invariant best-match searching in the subspace $\widehat{\Omega}$. The speed-up in the scheme results from several levels of pruning different distance computations:

1. *Pruning of rotation distance computations.* The previously derived property allows us to avoid computing a large percentage of the RD distances between the query Q and the elements of the dataset $\widehat{\Omega}$.

2. *Pruning of inner distance computations.* As the inner distance D also satisfies the triangle inequality, for every time series C_i that was not pruned on the previous level, only part of the inner distances between Q and the rotated versions of C_i need to be computed.

3. *Pruning of primitive distance operations.* Using a simple technique, called *early abandon* (to be described later), one can further speed up the inner distance computations that were not pruned in the previous step, by skipping some of the primitive pairwise computations between the scalar elements of the compared time series.

All three levels contribute to the speed-up of the nearest neighbor searches in the rotation invariant space, but it is the pruning of rotation distance computations that becomes of particular importance especially as the dataset size grows very large. While the pruning of inner distance computations and primitive operations still requires that all m dataset time series are retrieved, the pruning of rotation distances makes it feasible for comparing only part of them. This makes the simple result from the previous section extremely important for cases when disk retrievals and indexing are required.

4.2.1 Best-Match Search Algorithm

The proposed scheme is an adaptation of Burkhard-Keller's fast file searching algorithm described in [6]. Here we assume that all rotation distances from the elements of $\widehat{\Omega}$ to a preselected *center point* C_r (see Section 4.2.3) are computed and stored in a sorted list RL. We also precompute the *self-distances* between C_i and its rotations and store them in a sorted n-dimensional vector DL_i, i.e. $DL_i = \{D(C_i^1, C_i^j)\}$, $\forall j \in [1..n]$ (Note that we do not store the n by n $\mathbf{C}_i$ matrices, but just the distance vectors). Maintaining all m self-distance vectors is necessary for the second level inner distances pruning and increases twice the memory requirement for the proposed scheme compared to the simple brute force search. This linear increase in space complexity is a reasonable and acceptable overhead, as it refers to the compact 1D time series representation rather than the 2D original images. The pseudo code with a detailed explanation of the rotation invariant searching is presented as Algorithm 1.

Algorithm 1 Rotation invariant best-match search

Preprocessing:

1: $C_r \in \widehat{\Omega}$ - preselected center
2: $RL = \{RD(C_r, C_i)\}$ - sorted list, $i \in [1..m]$
3: $DL_i = \{D(C_i^1, C_i^j)\}$ - sorted lists, $i \in [1..m],\ j \in [1..n]$

Search:

4: $\forall Q$: $[bm\_Q,\ Min\_Dist] = RI\_Search(\widehat{\Omega}, C_r, RL, RD)$

procedure $[bm, \xi]$=*RI_Search(Cnd, C, L, d)*
in: *Cnd*: candidates; *C*: center; *L*: list distances; *d*: dist func
out: *bm*: best-match; ξ: minimal distance

5: $\xi = d(C, Q)$
6: $bm = C$
7: $Cnd = Cnd \setminus C$
8: **while** $Cnd \neq \varnothing$ **do**
9: select $C_i \in Cnd$:
$|d(C_i, C) - d(Q, C)| \leq |d(C_j, C) - d(Q, C)|$,
$\forall C_j \in Cnd, i \neq j$
10: **if** $d = RD$ **then**
11: $[bm_{fake}, \xi_{tmp}] = RI\_Search(\mathbf{C}_i, C_i^1, DL_i, D)$
12: **else**
13: $\xi_{tmp} = EarlyAbandon(C_i, Q, \xi)$
14: **end if**
15: **if** $\xi_{tmp} < \xi$ **then**
16: $\xi = \xi_{tmp}$
$bm = C_i$
17: **end if**
18: $\forall C_l \in Cnd\ \wedge\ |d(C_l, C) - d(Q, C)| > \xi$:
$Cnd = Cnd \setminus C_l$
19: **end while**

For clarity of presentation the described algorithm returns only the best-match to a given query. The extension finding the k most similar time series to the query is straightforward.

For every incoming query the search routine *RI_Search* is invoked with: a best-match candidates list *Cnd* initialized as the whole dataset $\widehat{\Omega}$; the list RL of the precomputed distances from all dataset elements to the center point; and the type of distance function set to RD. The distance function is also used to differentiate between the first and second levels of pruning. *RI_Search* sets the initial best-match element *bm* to the center point and the current minimal distance ξ to the distance between the query and the center. The iterative search of the candidates list then proceeds in three steps. While the list is not empty, a new candidate is selected (line 9), using the heuristic suggested by Burkhard and Keller (described below). If the distance to the new candidate is smaller than the current minimal distance, then the best-match so far is updated to the new candidate (line 16). Finally, the triangle inequality is applied (line 18) to prune all candidates that are guaranteed to be further from the query than ξ. More precisely, as $d = RD$ or D which both satisfy the triangle inequality, the following two inequalities hold: $d(Q, C_l) + d(Q, C) \geq d(C_l, C)$ and $d(Q, C_l) + d(C_l, C) \geq d(Q, C)$, or in a more compact form $d(Q, C_l) \geq |d(C_l, C) - d(Q, C)|$. Therefore, if the difference of the already computed $d(C_l, C)$ and $d(Q, C)$ is larger than the currently minimal distance ξ, then the distance from the query to the candidate C_l is guaranteed to be also larger than ξ, and there is no need to explicitly compute it.

For the first step, the candidate selection, Burkhard and Keller suggest choosing an element C_i still in the candidates list, for which the difference $|d(C_i, C) - d(Q, C)|$ is minimal (line 9). Note that this difference is a lower bound for the distance $d(Q, C_i)$, thus it is likely that by choosing the element with the minimal difference we also choose an element that is closer to the final solution. In the experimental evaluation we found out that the heuristic is essential for the pruning capability of the algorithm and its faster convergence to the solution. As the distance list L is sorted, the first candidate can be selected in logarithmic time. Suppose the binary search for a candidate shows that $d(C_i, C) < d(Q, C) < d(C_{i+1}, C)$, where i corresponds to the position of $d(C_i, C)$ in the sorted list L. This means that the heuristic will return as candidate either C_i or C_{i+1}. On subsequent iterations, one does not need to perform the binary search again but rather select the candidate that is still in the candidates list and whose distance to the center is closest to $d(Q, C)$ in either direction left or right.

When the algorithm is invoked with the rotational distance RD as distance function, i.e. we are on the first level of pruning rotation distances, the selected candidate C_i needs to be rotated n times and the inner distances $D(C_i^j, Q), j \in [1..n]$ need to be computed. We can do this again by applying the *RI_Search* procedure (line 11, second pruning level), this time with a center C_i^1, sorted list of distances to the center DL_i, and the inner distance D as a distance function. The list of candidates is now composed of every rotation of C_i, which is simply the rotation matrix $\mathbf{C}_i$. When a candidate C_i^j for an inner distance computation is identified, the actual inner distance $d(Q, C_i^j)$ can be optimized further by computing it with an early abandon technique (line 13, third pruning level).

4.2.2 Early Abandon

The early abandoning is a simple, yet extremely efficient technique for speeding up the computations of a distance function. In Section 5.2 we show that the running time of the brute force search can be improved with more than a factor of two, by simply modifying it with an early abandon criterion. The method uses a threshold ξ_t and computes the inner distance D by accumulating the primitive pairwise distances as long as the sum is smaller than the threshold. If the threshold is reached, the computation of the inner distance is abandoned. For completeness of the presented searching scheme we list the early abandon method below,

for the case when the inner distance is an L_p-norm.

Algorithm 2 Early abandon for L_p-norms

procedure ξ=*EarlyAbandon*(C, Q, ξ_t)
in: $C = (c_1, \ldots, c_n)$; $Q = (q_1, \ldots, q_n)$; ξ_t: threshold
out: ξ: $L_p(C, Q)$ terminated by threshold ξ_t
1: $\xi = 0$; $\xi_t = (\xi_t)^p$; $i = 1$
2: **while** $(\xi < \xi_t) \wedge (i \leq size(C))$ **do**
3: $\quad \xi \mathrel{+}= |c_i - q_i|^p$
4: $\quad i \mathrel{+}= 1$
5: **end while**
6: $\xi = \sqrt[p]{\xi}$

The algorithm is specific for the distance function used, as different functions might pair different scalar elements. For example in the case of Dynamic Time Warping, c_i might be paired with q_j $(i \neq j)$. Still, an equivalent early abandoning cut-off criterion can be applied to prune some of the paths in the dynamic programming matrix used by the DTW algorithm.

4.2.3 Center Selection

The percentage of distance computations that are excluded from consideration, and thus the performance of the algorithm, is higly dependent on the pruning capability of the selected center point C_r. In the original Burkhard-Keller algorithm, the selection is made at random. There are two factors that determine how good C_r is, namely, its position in the subspace $\widehat{\Omega}$ with respect to the other dataset points, and its position with respect to the queries. A suitable center point will have a small difference $|RD(C_j, C_r) - RD(Q, C_r)|$ for just a few dataset points C_j. Shapiro [19] argues that good centers can be points, which are further from the center of any cluster that might be present in the dataset. This is so, because points close to the cluster centers will be in close proximity to many other points, and for most of those neighbors the above difference will be small. Shapiro suggests an extension of Burkhard-Keller's searching algorithm in which k random centers, rather than one, are used. While this has the potential to mitigate the effect of choosing one inappropriate center, it also comes at the cost of increasing the memory requirements k times, if we would like to apply it for the second level of inner distance pruning too.

In our implementation we still use a single center point, but rather than randomly selecting it, we use subsampling. For the purpose, the preprocessing step (line 1, Algorithm 1) is modified as follows. A small training and validation subsets, are randomly selected from $\widehat{\Omega}$. The center C_r is set to the point from the training subset that has the best pruning capability for the queries from the validation set. Using subsampling in the center selection process implicitly takes into consideration the specific data distribution, which leads to better pruning ability and smaller variance as compared to random center selection. If, however, the actual distribution of the queries is unknown, i.e. it differs from the distribution of the dataset at hand, then a random or multiple centers should be considered.

The above preprocessing is performed only for the first pruning level. For the second pruning level we always use as centers the original series, i.e. C_i^1. Still, as seen from the evaluation in Section 5.2, the variance in the performance is very small, which suggests that any rotation C_i^j is an equally suitable center. An intuition of the phenomenon is provided by the observation that for L_p-norms the following equality is true: $D(C_i^{j_1}, C_i^{(j_1+k) \bmod (n)}) = D(C_i^{j_2}, C_i^{(j_2+k) \bmod (n)})$, $j_1, j_2, k \in [1..n]$. The fact implies that every rotation C_i^j is distributed in the same way among the rest of the rotations of C_i. The small variation in the performance results from the difference in the mutual positions of the query and the different rotations, but on average every rotation will have similar pruning power.

5. EXPERIMENTAL EVALUATION

The performance of the *RI_Search* algorithm is evaluated, utilizing the ubiquitous Euclidean distance as inner metric. First, the effectiveness of the simple 1D representation is demonstrated by comparing its accuracy to previously published results. The speed-up introduced by the presented approach is then discussed for three shape datasets[2], each exhibiting different properties. To illustrate the contribution of the individual pruning levels, a break-up of the improvement into components has also been provided.

5.1 Representation Effectiveness

In general this paper is not making any claims about the effectiveness of shape matching using time series representation. Because we are simply speeding up arbitrary distance calculations on arbitrary 1-dimensional representations of shapes, we automatically inherit the well-documented effectiveness of other researchers published work [1, 2, 3, 9, 12, 21]. Nevertheless, for completeness we demonstrate the effectiveness of the adopted shape representation on several datasets. Table 2 shows the error rate of one-nearest neighbor classification as measured using leaving-one-out evaluation.

Table 2: The Classification Error of Euclidean distance with time series extracted from all boundary points

Name	Classes	Instances	Euclidean Error
Face	16	2240	3.839%
SwedishLeaf	15	1125	13.33%
Chicken	5	446	19.96%
MixedBag	9	160	4.375%
OSU Leaves	6	442	33.71%
Diatoms	37	781	27.53%

Recall that Euclidean distance has no parameters, once the time series are extracted. For the *Face* and leaf datasets the (approximate) correct rotation was known. We removed this information by randomly rotating the images. The *MixedBag* dataset is small enough to run the more computationally expensive Chamfer [5] and Hausdorff [16] distance measures. They achieved an error rate of 6.0% and 7.0% respectively (see also [21]), slightly worse than Euclidean distance. Likewise the *Chicken* dataset allows us to compare directly to [15], which 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%. For the *Diatom* dataset, the results are competitive with human experts, whose error rates ranged from 57% to 13.5% [12], and only slightly worse than the

[2]All datasets used in the evaluation are publicly available and can be downloaded from the following URL: xxx.

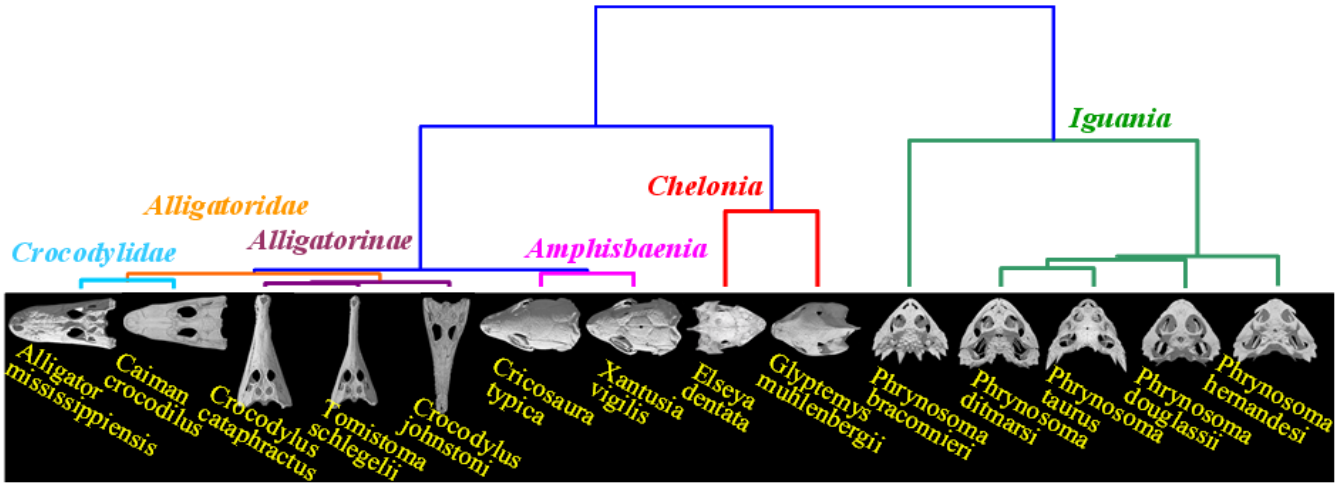

Figure 3: Hierarchical clustering of fourteen reptile skulls using the adopted representation.

Morphological Curvature Scale Spaces (MCSS) approach of [12], which got 26.0%. Note, however, that the MCSS has several parameters to set.

We also performed extensive "sanity check" experiments using a large database of reptile skulls. We performed a hierarchal clustering and compared the results with the current consensus on reptilian evolution as suggested by DNA evidence [10, 11]. Figure 3 shows a typical example. Two things should be noted: the clustering is subjectively sensible and clearly is rotation invariant. Furthermore, while the global clustering does not perfectly agree with the evolutionary consensus, all the major groups are clustered together as we have annotated in Figure 3. In other words, the shape measurements do produce high quality morphological phenograms, although convergent evolution prevents us from obtaining the true global phylogenetic tree from just an examination of skulls.

5.2 Performance Improvement

Three publicly available datasets are used in the efficiency evaluation of the *RI_Search* algorithm - *Arrowheads*, *SQUID*, and *Heterogeneous* shapes. The goal was to test the expected average improvement with respect to several factors: when the space density (i.e. the dataset size m) is varied; when the dimensionality of the space n is varied; and when the space is composed of elements from several largely separable and comparatively dense clusters.

All experiments represent averages over 50 randomly drawn queries, that are subsequently removed from the dataset. For correctness we show both the percentage improvement in terms of primitive operations and in terms of running times. There is not a one-to-one mapping between the two results, due to implementation and language specific biases, which allow for certain constructs to be executed in a more optimized manner than others.

ARROWHEADS Dataset. The dataset represents a large collection of arrowheads with various shapes and sizes. Figure 4 depicts some representative classes from the data.

We have further augmented the dataset with new images by scaling, deforming and rotating some of the original shapes. The overall size of the resulting dataset is 15000

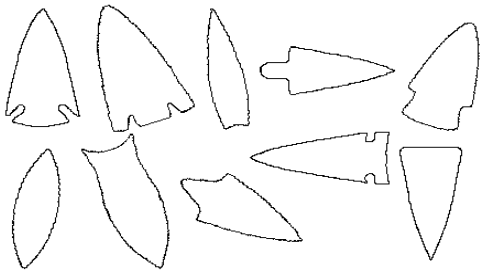

Figure 4: Representative examples from the *Arrowheads* dataset.

samples. After extracting the time series from the shapes, we resample them to $n = 250$ time points, which for this dataset seems to preserve the accurate representation. Prior to storing, all resampled time series have further been normalized to have a mean zero and standard deviation one.

The percentage improvement of our approach over the *BruteForce* search in terms of performed primitive distance computations is illustrated on Figure 5. The performance of simply applying the *EarlyAbandon* technique is also included for comparison.

There are several important aspects to observe in the result. Increasing the space density, i.e. introducing more samples in the dataset, increases the pruning power of the algorithm. The effect is expected as with more elements the chance of finding a sample that is very close to the query is higher. Such samples minimize significantly the cut-off threshold ξ, and a lot of the remaining elements start failing the triangle inequality test. The same is true for the *EarlyAbandon* cut-off criterion. Still, the *RI_Search* algorithm performs far less operations than *EarlyAbandon* - more than two times less operations for the smallest dataset size (5.48% - *RI_Search* vs 12.04% - *EarlyAbandon*), and more than twenty times less operations for the largest dataset size (0.19% - *RI_Search* vs 3.88% - *EarlyAbandon*). For all dataset sizes of 500 elements and above the *RI_Search* algorithm performs less than 1% of the operations performed by the exhaustive *BruteForce* search algorithm.

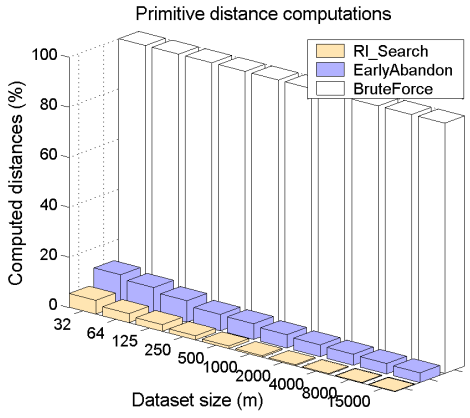

Figure 5: *Arrowheads*. Expected percentage of distances to be computed as compared to *BruteForce* search.

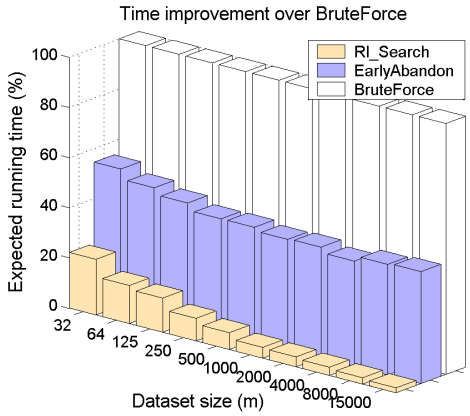

Figure 6: *Arrowheads*. Expected running time as compared to *BruteForce* search.

As previously noted, the time improvement does not correlate exactly to the operations improvement because of language and implementation specifics (see Figure 6). Even though *EarlyAbandon* executes less than 10% of all primitive operations, in our implementation it hardly speeds up the search algorithm more than twice for any value of m. We believe this results from the fact that the time for accessing all training samples and their rotations dominates the time for loop computations over array structures as executed by the language. The time improvement for the *RI_Search* algorithm is also smaller than the operations improvement, which is due mainly to overheads from supporting the sorted candidates and distances lists. Additional, very small slowdown is also caused by the binary search of the first candidate and the traversal of the lists for excluding candidates that fail the triangle inequality. Overall, the proposed algorithm is from four ($m = 32$) to more than fifty ($m = 15000$) times faster than the *BruteForce* search. On a Pentium4 $3GHz$ processor, for $m = 15000$ elements, the average time of *RI_Search* to find the best rotation invariant match to a query is 0.2 sec.

It is important to understand how much each pruning component contributes for the final operations improvement introduced by the algorithm. Fewer computations of the rotation distance imply accessing fewer shapes, which is essential especially when indexing is applied. And fewer inner distances to be considered suggest less memory accesses to different elements, which as we saw from the time performance of *EarlyAbandon*, is also of primary importance. Table 3 gives a break-up for *RI_Search* into levels of pruning.

Table 3: Percentage of performed operations. *Row1*: Percentage of computed rotation distances. *Row2*: Percentage of computed distances out of all possible inner distances after level one was performed. *Row3*: Percentage of primitive operations out of all possible remaining operations after level two was performed

Pruning Level	Mean(Deviation) of Performed Operations(%)			
	$m = 250$	$m = 500$	$m = 1000$	$m = 15000$
1-st	52.1(12.3)	45.5(12.7)	34.1(10.0)	22.7(5.81)
2-nd	19.9(0.85)	17.7(0.60)	15.5(0.79)	9.81(0.31)
3-rd	16.0(0.91)	12.7(0.84)	11.4(0.38)	8.61(0.30)

The table illustrates how powerful the triangle inequality could be, especially for larger datasets. For example, when $m = 15000$ the algorithm avoids examining almost 80% of the shapes. The second and the third pruning levels are presented with respect to the possible operations after the previous pruning level has been performed. For example, after eliminating some rotation and inner distances, the early abandoning subroutine executes 8.61% of the remaining primitive operations. This number is twice higher than applying directly *EarlyAbandon* on the initial data (3.88%, see Figure 5, $m = 15000$). The effect is expected as the time series that remain after the first two pruning levels are all closer to the query, and on average the *EarlyAbandon* cut-off criterion is not that efficient for them.

Finally, the standard deviation in the performed operations for each pruning level is also presented in the table. The small variance in the second pruning level suggests that all rotated versions of a time series C_i are equivalent in a certain sense, as the rest of the rotated series are similarly distributed around them. Therefore, as discussed in Section 4.2.3, any rotation C_i^j can be considered an equally suitable choice for an inner distance center.

SQUID Dataset. The dataset contains 1100 images of marine creatures. Figure 7 demonstrates several samples from the database.

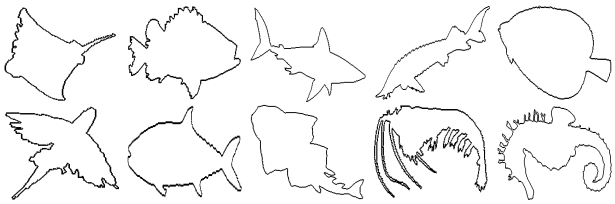

Figure 7: Representative examples from the *SQUID* dataset.

The shapes are preprocessed as described for the *Arrowheads* dataset. Most of the creatures have horizontal orientation, still as the figure shows there are samples that have arbitrary orientation too. For this dataset we use the original shapes without any further transformations. All extracted time series are resampled to $n = 1000$ time points. The higher dimensionality suggests a more sparsely populated

space, which is the reason for the worse expected improvement compared to the *Arrowheads* dataset (see Figure 8)

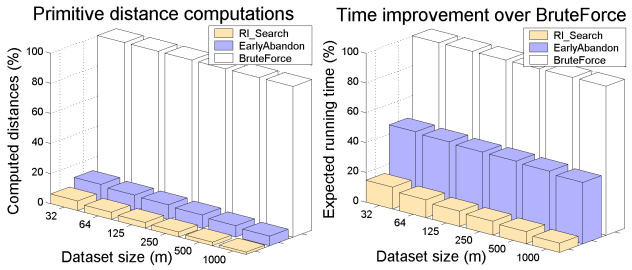

Figure 8: *SQUID*. Improvement over *BruteForce* search. *Left*: Expected percentage of primitive operations to be performed. *Right*: Expected running time.

For example, the total percentage of operations to be performed are: 1.60% - *SQUID* vs 0.60% - *Arrowheads* for $m = 1000$; 2.34% - *SQUID* vs 1.03% - *Arrowheads* for $m = 500$ etc. The difference is due almost entirely to the larger percentage of rotation distances that need to be compared now as seen in Table 4. The table shows that for the same dataset sizes the percentage of computed rotation distances for *SQUID* shapes is twice higher than the one for *Arrowheads* shapes. The percentage of computed inner distances, on the other hand, remains approximately the same.

Table 4: *SQUID*. Percentage of performed operations.

Pruning Level	Mean(Deviation) of Performed Operations(%)			
	$m = 125$	$m = 250$	$m = 500$	$m = 1000$
1-st	72.2(9.73)	71.2(11.1)	66.1(11.3)	59.2(11.6)
2-nd	20.1(1.32)	19.2(1.70)	17.3(1.41)	15.4(1.46)
3-rd	25.9(0.90)	23.8(1.50)	20.5(0.36)	17.6(0.35)

Similarly to the *Arrowheads* dataset, increasing the number of time series leads to a linear increase in the pruning capability of the *RI_Search* algorithm. The average query time now is much longer, e.g. for $m = 1000$ we obtained 0.9 sec for the *SQUID* vs 0.04 sec for the *Arrowheads* dataset. Nevertheless, the overall speed-up introduced by *RI_Search* is preserved. In terms of running time, for $m = 1000$ the algorithm is more than sixteen times faster than *BruteForce* and more than six times faster than *EarlyAbandon*(see Figure 8, *Right*).

HETEROGENOUS dataset. To test the pruning ability of the *RI_Search* algorithm when the data come from different distributions and represent several distinct clusters, we have combined images from five datasets into a single collection of size $m = 5850$.

The number of elements in each of the five clusters is approximately the same and each of them can be divided further into several subclusters. The datasets included are *Arrowheads* (described earlier in this section), *Diatoms*, *Chicken*, *MixedBag* and *SwedishLeaf* (see Figure 9). The shapes are preprocessed as described earlier for the *Arrowheads* dataset. All extracted time series have been resampled to $n = 1024$ time points, i.e. the space is of similar dimensionality as for the *SQUID* dataset. Figure 10 shows the expected improvement introduced by the *RI_Search* algorithm.

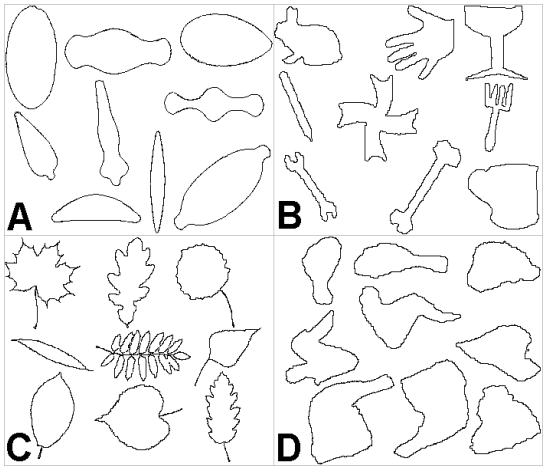

Figure 9: Representative examples from the *Heterogenous* dataset.The collection includes samples from five datasets: *Arrowheads*, *Diatoms* - **A**, *MixedBag* - **B**, *SwedishLeaf* - **C** and *Chicken* - **D**

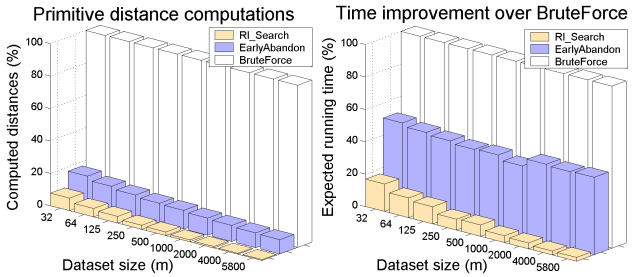

Figure 10: *Heterogenous*. Improvement over *Brute Force* search. *Left*: Expected percentage of primitive operations to be performed. *Right*: Expected running time.

Though the overall performance is comparable to the one observed for the *SQUID* dataset, having five largely separable clusters introduces certain specifics too. For example, for a random query drawn from any of the subsets, the *RI_Search* algorithm eliminates relatively quickly the training samples that come from the other four datasets. Therefore, the number of computed rotation distances for the corresponding dataset sizes now is much smaller (see Table 5).

Table 5: *Heterogenous*. Percentage of performed operations.

Pruning Level	Mean(Deviation) of Performed Operations(%)			
	$m = 250$	$m = 500$	$m = 1000$	$m = 5800$
1-st	31.4(5.98)	35.3(8.27)	24.3(6.39)	25.3(9.25)
2-nd	32.9(1.76)	28.9(1.82)	25.6(1.73)	17.1(1.91)
3-rd	24.2(1.08)	21.6(1.39)	19.5(1.07)	12.6(1.08)

The inner distance computations, on the other hand, are more as opposed to the inner distances computed for the *SQUID* shapes. The reason for the worse pruning of inner distance computations is that many of the shapes are approximately symmetric with respect to both of their major axes. This is true for shapes from different dataset, e.g. *Diatom*, *Arrowheads* or *SwedishLeaf*. The worst case for the

inner distance pruning is when the shape is approximately spherical (some of the *Diatom* elements), but even with symmetry only on the axes there are still many rotations that are identical, which makes the inner distance cut-off criterion not so efficient.

6. CONCLUSIONS AND FUTURE WORK

The work demonstrates that, under certain conditions, rotation invariant distance measures define a metric over the shape space, which implies that searching in this space could be highly optimized. We presented an algorithm that exploits this observation and speeds up some existing best-match searching approaches, by avoiding large number of the distance computations.

The obtained results raise some interesting questions and point out important, from machine learning and data mining perspective, directions for possible extensions. For example, though a generalization of the algorithm for $3D$ shapes seems straight forward, it is not obvious how badly the increase in dimensionality will impact the expected performance improvement. Another significant direction to follow is the adaptation of the algorithm for other computationally intensive tasks, as clustering or classification, when rotation invariance is required. We are actively exploring both of the above directions.

7. ADDITIONAL AUTHORS

8. REFERENCES

[1] C. Adamek and N. O'Connor. A multiscale representation method for nonrigid shapes with a single closed contour. *IEEE Circuits and Systems for Video Technology*, 14(5):742–753, 2004.

[2] T. Adamek and N. O'Connor. Efficient contour-based shape representation and matching. *Multimedia information retrieval*, pages 138–143, 2003.

[3] E. Attalla and P. Siy. Robust shape similarity retrieval based on contour segmentation polygonal multiresolution and elastic matching. *Pattern Recognition*, 38(12):2229–2241, 2005.

[4] B. Bhanu and X. Zhou. Face recognition from face profile using dynamic time warping. pages IV: 499–502, 2004.

[5] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, 1988.

[6] W. Burkhard and R. Keller. Some approaches to best-match file searching. *Commun. ACM*, 16(4):230–236, 1973.

[7] A. Cardone, S. Gupta, and M. Karnik. A survey of shape similarity assessment algorithms for product design and manufacturing applications. *J. Comput. Inf. Sci. Eng.*, 3(2):109–118, 2003.

[8] K. Fukunaga and P. Narendra. A branch-and-bound algorithm for computing k-nearest neighbors. *IEEE Trans. Comp.*, 24(7):750–753, 1975.

[9] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1312–1328, 1999.

[10] W. Hodges and K. Zamudio. Horned lizard (*p*hrynosoma) phylogeny inferred from mitochondrial genes and morphological characters: understanding conflicts using multiple approaches. *Molecular Phylogenetics and Evolution*, 31:961–971, 2004.

[11] N. Iwabe. Sister group relationship of turtles to the bird-crocodilian clade revealed by nuclear DNA-coded proteins. *Molecular Biology and Evolution*, 22:810–813, 2004.

[12] A. Jalba, M. Wilkinson, J. Roerdink, M. Bayer, and S. Juggins. Automatic diatom identification using contour analysis by morphological curvature scale spaces. *Machine Vision and Applications*, 16(4):217–228, 2005.

[13] S. Kim, S. Park, and W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *ICDE*, pages 607–614, 2001.

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

[15] R. Mollineda, E. Vidal, and F. Casacuberta. Cyclic sequence alignments: Approximate versus optimal techniques. *Approximate versus optimal techniques. International Journal of Pattern Recognition and Artificial Intelligence*, 16(3):291–299, 2002.

[16] C. Olson and D. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *IEEE Transactions on Image Processing*, 6(1):103–113, 1997.

[17] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4):807–832, 2002.

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

[19] M. Shapiro. The choice of reference points in best-match file searching. *Commun. ACM*, 20(5):339–343, 1977.

[20] R. Veltkamp and L. Latecki. Properties and performance of shape similarity measures. *IFCS Conference: Data Science and Classification*, 2006.

[21] M. Vlachos, Z. Vagena, P. Yu, and V. Athitsos. Rotation invariant indexing of shapes and line drawings. In *Proceedings of the 14th ACM international conference on Information and knowledge management(CIKM)*, pages 131–138, 2005.

[22] Z. Wang, Z. Chi, D. Feng, and Q. Wang. Leaf image retrieval with shape features. In *4th International Conference on Advances in Visual Information Systems*, pages 477–487, 2000.

[23] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *4th annual ACM-SIAM Symposium on Discrete algorithms (SODA)*, pages 311–321, 1993.

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

[25] J. Zunic, P. Rosin, and L. Kopanja. Shape orientability. In *ACCV(2)*, pages 11–20, 2006.