Efficient Rotation Invariant Retrieval of Shapes with Applications in Medical Databases

Selina Chu[1], Shrikanth Narayanan[1,2], and C.-C. Jay Kuo[1,2]
[1]*Department of Computer Science,* [2]*Department of Electrical Engineering*
University of Southern California, Los Angeles, CA 90089
{selinach, shri, cckuo}@sipi.usc.edu

Abstract

Recognition of shapes in images is an important problem in computer vision with application in various medical problems, including robotic surgery and cell analysis. The similarity measures for such purpose must be robust to various transformations and modest occlusions. Transformations, such as scaling and translation can be handled easily by techniques through data representations or similarity measures. Rotation invariance is an inherently more difficult problem and can be handled through data representation, but at the expense of poor discrimination. Approaches which provide excellent discrimination require a complexity of $O(n^3)$ for each shape comparison. In this paper, we present a framework that provides a speedup over the slow but accurate approaches. The algorithm is inspired by the iterative deepening framework in artificial intelligence, by examining the data at increasingly fine levels of approximation until it is either considered irrelevant or submits to the full calculations. Although we examine the data several times at different levels of abstractions, because the time required for the last iteration dwarfs all others, this apparent redundancy is inconsequential. We will show that our method provides at least a 3-4 orders of magnitude in speedup without generating any false dismissals.

1. Introduction

Recognition of shapes in images is an important problem in computer vision with applications in medical domains, including robotic surgery and cell analysis. There is a growing interest in the medical community in efficient content-based image retrieval and analysis of biomedical images [2, 3]. Medical experts have found shapes to provide a consistent and reliable way for identifying various pathologies from an image collection [2]. Another use of shape matching can be found in the study of cells [4]. The use of direct comparison of images is intractable for large datasets due to the high dimensionality of raw data. In addition, idiosyncrasies in the details of actual images may be irrelevant to the matching process. Therefore, a fast recognition of contours provides an efficient and robust way of accelerating the search. An extensive survey of feature extraction and shape matching techniques can be found in [5, 6]. The similarity measures for such shape matching must be robust to various transformations and modest occlusions. Transformations, such as scaling and translation, can be handled easily by techniques through data representations or similarity measures. Unlike the other two transformations, rotation invariance is inherently more difficult. Although rotation invariance can be handled through data representation, it is achieved at the expense of poor discrimination.

Approaches which provide excellent discrimination that utilize Dynamic Time Warping (DTW) or other dynamic programming algorithms have a high order of complexity [1]. The matching process in [1] involves the need to determine the best correspondence, or rotation, between contour points. This rotation invariance of the contour can be checked by using N circular shifts between representations in their matching process, where N is the length of each contour, represented as a time series. Further, DTW is used as their distance measure between contour points. Since DTW has a complexity of $O(N^2)$ and N circular shifts are required, the resulting complexity is $O(N^3)$ for each shape comparison.

Previously, we proposed an algorithm that utilized an iterative search method based on examining different levels of approximation to expedite time series retrieval using DTW [7]. Since its introduction, others have utilized this technique for a variety of applications, such as query-by-content [8, 9], melody retrieval [10], and speech recognition [11]. In this paper, we will demonstrate that we can use similar method for shape matching with rotation invariance, which will provide a speedup over the slow but accurate approaches. Unlike previously proposed approximate methods, we present here a framework that delivers similar speedup, with a guarantee of no false dismissals.

We begin by converting the shape into time series, as shown in Figure 1. The algorithm then begins to examine the data at increasingly fine levels of approximation until it is either admissibly pruned or is submitted to the full calculations. At each level, we use a lower bounding method to examine the data several times. Because the time required for the last iteration dwarfs all others, this apparent redundancy is inconsequential, and *most* comparisons will terminate after examining the data only at coarse levels of

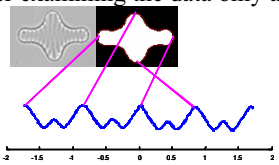

Figure 1: Shapes are converted to time series

Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)
0-7695-2517-1/06 $20.00 © 2006 IEEE

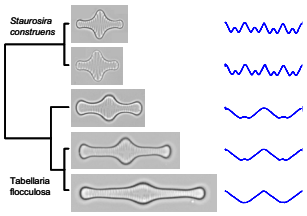

Figure 2: Five Diatoms from two different taxa, clustered together based of the shape of the extracted time series

approximation. We will demonstrate that our method provides at least a 3-4 orders of magnitude in speedup without generating any false dismissals.

2. Background

In this section, we will review the classic dynamic time warping (DTW) algorithm and its lower bounding measure.

2.1. Review of the dynamic time warping algorithm

Suppose we have two time series Q and C, of length n and m respectively, where:

$$Q = q_1, q_2, \ldots, q_i, \ldots, q_n \quad (1)$$

$$C = c_1, c_2, \ldots, c_j, \ldots, c_m \quad (2)$$

To align these two sequences using DTW, we construct an n-by-m matrix where the (i^{th}, j^{th}) element of the matrix contains the distance $d(q_i,c_j) = (q_i - c_j)^2$, aligning two points q_i and c_j. There are exponentially many warping paths, however we are interested only in the path which minimizes the warping cost:

$$DTW(Q,C) = \min\left\{\sqrt{\sum_{k=1}^{K} w_k}\right\} \quad (3)$$

The k^{th} element of a warping path W is defined as $w_k = (i,j)_k$, where w_k is a matrix element that defines a mapping between Q and C. This optimal path can be found efficiently using dynamic programming by evaluating the following recurrence:

$$\gamma(i,j) = d(q_i,c_j) + \min\{\gamma(i\text{-}1,j\text{-}1), \gamma(i\text{-}1,j), \gamma(i,j\text{-}1)\} \quad (4)$$

where $\gamma(i,j)$ is the cumulative distance of $d(i,j)$, which is the distance found in the current cell, and the minimum of the cumulative distances of the three adjacent elements. To restrict the number of warping paths, several constraints are applied, including *boundary condition*, *continuity condition*, and *monotonicity condition*.

The time complexity of DTW is $O(nm)$. This review of DTW is necessarily brief; we refer the interested reader to [12, 13] for a more detailed treatment.

Due to the complexity of DTW, a global constraint is usually applied to the warping path, providing both a speedup and to prevent pathological warpings, where small section of one sequence maps onto a much larger section of another. Two of the most commonly used global constraints, or warping window, are the the Sakoe-Chiba Band [14] and the Itakura Parallelogram [15].

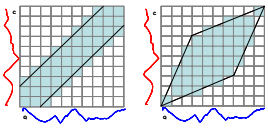

Figure 3: Example of global constraints: Sakoe-Chiba Band (Left), Itakura Parallelogram (Right)

2.2. Lower bounding for DTW

In this paper, we will use the lower bounding measure introduced by Keogh in [16], which enables DTW to be indexable. We use this lower bounding measure as a foundation for our rotation invariant problem. For a thorough explanation and proof of this lower bound, we refer the reader to the original paper. Instead, we will provide a brief explanation of this method.

The essence of this DTW lower bound lies in the use of the global constraints, or more commonly referred to as the warping window, illustrated in figure 4. This warping window creates a bounding envelope, enclosing the time series, Q.

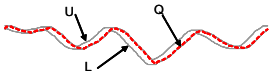

Figure 4: Sequences U and L created as envelope for Q

As described in section 2.1, the k^{th} element of a warping path W is defined as $w_k=(i,j)_k$, constrained to $j - r \le i \le j + r$, where r is the allowed range of warping. For the Sakoe-Chiba Band, r is a constant, and the Itakura Parallelogram, r is a function of i. Let us define two sequences U and L as the Upper and Lower of the envelope. This can be viewed as the amount of leeway that the time series Q can be warpped, regardless of where the warping path may lead itself based on the global constraint defined. U and L, in terms of r, are denoted as

$$U_i = \max(q_{i-r}, \ldots, q_{i+r}) \quad (5)$$

$$L_i = \min(q_{i-r}, \ldots, q_{i+r}) \quad (6)$$

An important property that U and L have is:

$$\forall i \quad U_i \ge q_i \ge L_i \quad (7)$$

Therefore, the lower bounding measure for DTW is given as follows:

$$LB\_DTW(Q,C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (c_i - U_i)^2 & \text{if } c_i > U_i \\ (c_i - L_i)^2 & \text{if } c_i < L_i \\ 0 & \text{otherwise} \end{cases}} \quad (8)$$

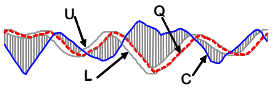

Figure 5: Lower bounding function for DTW – squared sum of the (gray lines) distances of C that is outside of the envelope

In short, the Euclidean distance of every part of the candidate sequence, C, not falling within this bounding envelope of the query Q, is returned as its lower bound. One of the requirements for this lower bound is the need for the sequences to be the same length. Therefore, if they are

different, one of them must be reinterpolated to match the other.

3. Rotation invariant matching process

Shapes, or closed contour of images, can be commonly represented as 1D representation. There are many techniques in literature for converting a shape of an image to time series. Due to space limitation, we will refer interested readers to [17, 18] for a more detailed explanation. For the rest of this paper, we will assume each shape to have a 1D representation, which we referred to as a time series. We can also assume that any of the time series can be transformed back to the original shape.

3.1. Searching for shapes in database

To match two shapes, we can simply find the distance measure between two time series, using any applicable distance measure, such as Euclidean distance or DTW. However this might produce poor result due to the rotations of the shapes. To determine the optimal rotation, we need to align two shapes in a way that minimizes the error between them. To achieve this, we can hold one shape fixed, while rotating the other, and then finding the distance between them for each rotation. We take the minimum distance among all rotations and its location as the optimal rotation and distance. As explained in [1], each rotation is regarded as a circular shift in the time series d with n elements, where $d = d_1, d_2, \ldots, d_{n-1}, d_n$, becomes $d' = d_n, d_1, d_2, \ldots, d_{n-1}$.

The brute force algorithm is given as follows:

```
Algorithm Brute_force_database_search (Q, D)
  best_so_far := inf;
  location_of_best_so_far := null
  For i in 1 … number of time series in D
    distance_i := True_distance_of _2shapes( Q, D_i )
    If distance_i < best_so_far
      best_so_far := distance_i
      location_of_best_so_far = i
    end if;
  end for;
return best_so_far, location_of_best_so_far
end Brute_force_database_search

Function True_distance_of_2shapes(Q, C)
  best_rot_so_far := inf
  For i in 1 … number of elements in C
    C' := circular shift one element in C
    dist := similarity measure (Q, C')
    If dist < best_rot_so_far
      best_rot_so_far := distance
    end if
  end for
  return best_rot_so_far
end True_distance_of_2shapes
```

3.2. Optimization within Euclidean distance

Because similarity measurement can be computationally expensive, we would like to optimize the calculations as much as possible. Euclidean distance (Eq. 9), the most common distance measure, is amiable to several optimizations.

$$D(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad (9)$$

One obvious optimization is derived from the observation that the square root function is monotonic, thus we can remove it from the calculation and achieve identical rankings/classifications/clusterings. Once the square root step has been removed, optimization for similarly searching becomes apparent. The code to calculate the squared Euclidean distance can be modified to break out the loop if the partially accumulated distance exceeds the smallest distance encountered thus far.

```
accumulated_distance := 0;
  for i in 1 .. length_of_time_series loop
    accumulated_dist := accumulated_dist + (q_i – c_i)^2;
    if (accumulated_dist > best_so_far) then
      break
    end if
  end for
```

As soon as the accumulated distance exceeds the *best_so_far*, we can terminate and discard the candidate from further consideration.

3.3. Optimization within DTW

Optimizing DTW can also be performed, but in a different way. The DTW algorithm in Sec. 2.1 uses an iterative process to fill up the distance matrix. The warping path W is only found after all the cells in that matrix are computed. To optimize the DTW algorithm, we can check every column (after the distance measurements for that column are all found) whether the minimum of that column exceeds the *best_so_far* value. The minimum of a column must be picked as part of warping path, based on Eq. 4 and the continuity constraint. If this is the case, we abandon the entire process completely, and discard the candidate. Since this is less intuitive than Euclidean distance, Figure 6 illustrates this optimization. For example, we only fill in the distances for the shaded cells (utilizing the Sakoe-Chiba band). The highlighted column depicts the distance cells for one point from A to five different points on B. If we take the minimum of that column, it will be at least where the warping path traverses to. Since the entire warping path is unclear at this point until the entire table is calculated, this method allows us to perform a quick check to determine if it is worth calculating the rest of the table. This optimization has shown to speedup the calculations by at least three folds.

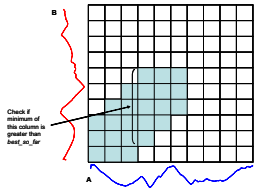

Figure 6: Illustration of optimizing DTW. Calculations can be abandoned when the minimum of a column in the distance matrix is found to be greater than *best_so_far*

4. Novel approach to rotation invariant matching

Section 3 introduces a brute force method for shape matching with rotation invariance. Despite the optimization with Euclidean distance and DTW, the process is still very lethargic, particularly for DTW. In this section, we introduce our framework that will expedite this search process. The intuition behind this algorithm is to iteratively examine

sequences at increasingly finer levels of approximation and compare the lower bounding results to the best match so far. Using this information, the algorithm iteratively considers whether the candidate sequence should either be dismissed as being unlikely to be a good match, or is worth considering at a yet finer level of approximation. We begin by reviewing the technique that enables us to approximate the sequences at various levels of reduced dimensions.

4.1. Dimensionality Reduction

We will use a dimensionality reduction technique that was independently introduced by [19] and [20] under different names; for clarity we will refer to it as Piecewise Constant Approximation (PAA). PAA can be found as follows:
Let us defined a time series of length n as $C=c_1, \ldots, c_n$. We can denote N to be the dimensionality of the space we wish to reduce n to, where $1 \le N \le n$. The time series C of length n can be represented in N dimensional space by the vector $\overline{C} = \overline{c}_1, \ldots, \overline{c}_N$. The i^{th} element of $\overline{C}$ is calculated as:

$$\overline{c}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} c_j \qquad (10)$$

In other words, PAA approximates a time series by dividing it into equal-length segments and recording the mean value of the data points that fall within each segment. This distance measure in the reduced space has been proven to always be less than or equal to the Euclidean distance of the original time series in [19]. An example of PAA is illustrated in part b in Figure 9.

4.2. Iterative deepening rotation invariant matching

For notational simplicity, we assume that the length n of the sequence in question is of a power of two. We can approximate a time series sequence using the PAA representation at any level of compression. Let us say, for example, the coarsest level is to approximate the sequences with 4 PAA coefficients. We call this level of approximation d_1 (for "depth 1"). If we need to calculate a more accurate approximation, we can double the resolution to 8 PAA coefficients, which we call d_2, etc. At increasing level, the approximation becomes finer and finer, until at $d_{\log2(n)-1}$, the approximation degenerates to the original data.

During the similarity search, we can use the lower bound to prune off unpromising candidates without performing the full and expensive circular DTW calculation. For now, let us assume the lower bound is a black box, which does not allow false dismissals. The lower bound for the non-rotation invariant DTW case has been explained in Section 2.2. But we will defer the details of the lower bounds for the rotation invariance case until the next two sections. The idea is that we perform the full DTW calculation on the first item in the database to initialize a *best_so_far* variable. Thereafter, when presented with a candidate sequence to test, we first test it using the coarse approximation of the query and candidate and their lower bound distance. If that distance is better than the *best_so_far*, we consider it a potential candidate and will continue onto a finer level, otherwise we discard that candidate immediately from further examination. Eventually as we compare the sequences at iteratively deeper levels of approximation, we will either prune away the candidate sequence or be forced to compare them at depth $d_{\log2(n)-1}$. At that level, there is no uncertainty about the calculated distance; if it is better than *best_so_far*, we update that value and continue to the next sequence.

4.3. Lower bounding for rotation invariance using Euclidean distance

Let us first examine the lower bounding measure for the rotation invariant case using Euclidean distance. We begin by creating a bounding range using approximations. Since we are interested in rapidly searching through the database to filter out irrelevant matches, we will create the upper and lower of the ranges using PAA. These are similar to Eq. 10; instead of finding the mean, we wish to acquire the min and max of these PAAs. Let us denote time series C with n elements as $C= c_1,c_2,\ldots,cj,\ldots,c_n$, where N is the dimensions we wish to reduce n to, where $1 \le N \le n$. Thus we have:

$$\overline{C}\_\max_i = \max\left(C_{\frac{n}{N}(i-1)+1}, \cdots, C_{\frac{n}{N}(i)}\right) \qquad (11)$$

$$\overline{C}\_\min_i = \min\left(C_{\frac{n}{N}(i-1)+1}, \cdots, C_{\frac{n}{N}(i)}\right) \qquad (12)$$

Figure 7 visualizes $\overline{C}\_\max_i$ and $\overline{C}\_\min_i$ as the upper and lower bound of the envelope around C.

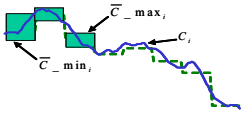

Figure 7: Sequences (*C_max* and *C_min*) used as limits for bounding envelopes using approximations.

We can now define the lower bounding function in the reduced dimension condition.

$$LB\_ED\_rotation(\overline{A},\overline{B}) = \sqrt{\sum_{i=1}^{n} \frac{n}{N} \begin{cases} (\overline{A}\_\min_i - \overline{B}\_\max_i)^2, & if \quad \overline{A}\_\min_i > \overline{B}\_\max_i \\ (\overline{B}\_\min_i - \overline{A}\_\max_i)^2, & if \quad \overline{B}\_\min_i > \overline{A}\_\max_i \\ 0 & otherwise \quad (overlap) \end{cases}} \qquad (13)$$

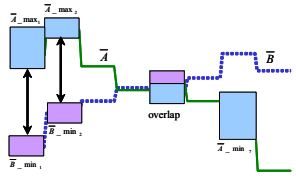

Figure 8: Lower bounding function for approximations using Euclidean distance

4.4. Lower bounding for rotation invariance using DTW

The lower bound for DTW is similar to the one denoted in Eq. 5 and 6. Since Eq. 5 and 6 is defined for the length of n, and n is too lengthy to allow efficient searching, especially with DTW, we need to first reduce the dimensionality to N

and then find the lower bound of that. To define the upper and lower bound of the envelopes in terms of PAAs, let us denote $\overline{C}$ to be PAA of time series C obtain from Eq. 10. For the uppers and lowers, we have

$$\overline{C}\_U_i = \max\left(\overline{C}_{i-r}, \cdots, \overline{C}_{i+r}\right) \quad (14)$$

$$\overline{C}\_L_i = \min\left(\overline{C}_{i-r}, \cdots, \overline{C}_{i+r}\right) \quad (15)$$

It is noted that, like Eq. 5 and 6, it also has the property of Eq. 7. Therefore, the lower bounding measure for DTW in the circular case is given as follows:

$$LB\_DTW\_rotation(\overline{Q}, \overline{C}) = \sqrt{\sum_{i=1}^{N} \frac{n}{N} \begin{cases} (\overline{C} - \overline{Q}\_U_i)^2, & if \quad \overline{C} > \overline{Q}\_U_i \\ (\overline{C} - \overline{Q}\_L_i)^2, & if \quad \overline{C} < \overline{Q}\_L_i \\ 0 \quad otherwise & (between\_U\_\&\_L) \end{cases}} \quad (16)$$

The lower bound is illustrated in the figure 10.

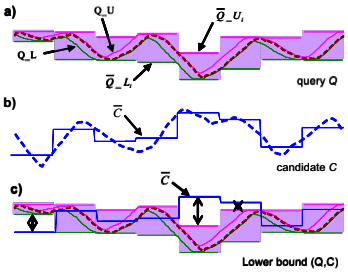

Figure 9: a) envelope of the query created by Eq. 11 and 12. b) the PAA of a candidate in the database. c) as we compare the distance between Q and C, we find the accumulated sum of the lengths of the black lines, squared-rooted, and then scaled by n/N, to be the lower bound.

4.5. Optimizing the iterative approach

The rotational positions among each contour provide us with an opportunity to further optimize the filtering process. We know the lower bound of a coarse approximation will never exceed that of a finer approximation. Although the overall distance of a potential candidate might be better than the *best_so_far*, not every position within that time series will produce such competitive results. Therefore, at each level, we eliminate the positions with results worse than the *best_so_far* and pass only the viable positions to the next level. We illustrate this intuition for optimizing the iterative process in Figure 10. Let us look at an example and assume that we have two time series of 256 points. We will consider three levels of approximation, from coarse to fine, at 32, 64, and 128 respectively. Each abscissa denotes one rotational position and each ordinate signifies the distance at that position.

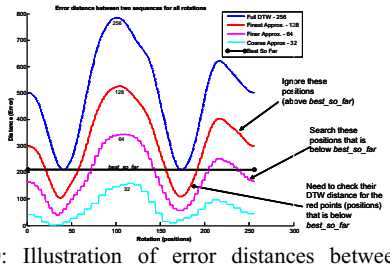

Figure 10: Illustration of error distances between each level. Optimization can be achieved by abandoning examination of positions that lies above the *best_so_far* line

There are 256 rotations between the two sequences. For all rotations at each approximation level, we decide whether it is below or above the *best_so_far*. We only need to exploit the rotational positions at the next level where the location yield results below the *best_so_far* line.

Putting everything together, we have the iterative deepening algorithm as follows:

```
Algorithm iterative_deepending_database_search (Q, D)
  best_so_far := inf;
  location_of_best_so_far := null
  N := initial amount of compression
  n := length of Q
  r := reach of warping path  (for DTW)
  for i in 1 … number of time series in D
    while N<n loop
      dist, positions:=get_lowerbound(Q, D(i), N, n, r, best_so_far, positions)
      If dist < best_so_far
        N:=2*N;
      else
        Break out of while loop
      end if
      If N==n
        dist := distance_of _2shapes( Q, D(i), positions ) %from section 3.1
        If dist < best_so_far
          best_so_far := dist
          location_of_best_so_far = i
        end if;
      end if
return best_so_far, location_of_best_so_far
end iterative_deepending_earch_database

Function get_lowerbound_dist(A, B, N, n, r, best_so_far, positions)
  A_PAA, B_PAA := eq. 10                                          {A}
  A_Max, B_Max := eq. 11 (for Euclidean dist), eq. 14 (for DTW)   {B}
  A_Min, B_Min := eq. 12 (for Euclidean dist) , eq. 15 (for DTW)  {C}
  for i in positions                                              {STEP 1}
    A_Max', A_Min' := Rotate to positions i
    LB1(i) := eq. 13 (for Euclidean dist), eq. 16 (for DTW)
  end for                                                         {end STEP 1}
  A' := Flip A;  %image is mirror-image
  Repeat {A, B, & C} for A'
  Repeat STEP 1 for new A_PAA, A_Max and A_Min to obtain LB2
  positions := all positions in LB1 and LB2 that are better than best_so_far
  return minimum of LB1 and LB2, positions;
End get_lowerbound_dist
```

The astute reader will note that, in the worst case, the iterative deepening method could reach the maximum depth each time. Naturally, this will be slower than the non-iterative approach. However, we are only looking at the lower bound of the approximation at each level, and this can be accomplished cheaply. Let us examine this more closely. We denote n to be the original length of the time series and N to be the dimension of the coarsest approximation, where $N << n$. We know that at the first level, the comparison of two shapes with circular shifts is $O(N^2)$. At each level, the approximation is double of the previous level. Let us further define C to be a constant number of times we have to multiply N for all levels of approximation. As noted previously, $O(n^3)$ is defined for using DTW with circular shifts. Therefore, the worst case for our iterative method comparing two shapes would be $O(CN^2)+O(n^3)$. In practice, the algorithm rarely has to explore sequences all the way to n. Even without the optimization described above, the amortized complexity over the entire database is $O(CN^2)+x(On^3)$, where x is some fraction of the size of the database. Since we only utilize the DTW algorithm on a fraction of the database, a dramatic speedup is observed.

The proof of the lower bound for DTW guaranteeing no false dismissals is given in [16]. Since we are utilizing the ranges of the approximation as their bounding envelopes, this creates an even weaker (and therefore also admissible) bound between the candidate and query sequence. We also know that the lower bounding distances can never be greater than the distance between the original sequences. Since we are

always comparing the lower bounds at the same rotational position and this is true for the case without rotations, the guarantee for no false dismissals also holds for all rotations.

5. Experimental Evaluation

In this paper, we are interested in the speedup obtained over the competitive algorithm for rotation invariant shape matching. Although the quality of the recognition is important, it will not be our focus. Instead we will demonstrate the effectiveness and improvement in speed. We test our algorithm on two publicly available datasets [21], leaves (Data1) and chicken (Data2), with 15 and 5 classes respectively. Both are of length 1024. All the data, including the query sequences, are normalized to mean of zero and a standard deviation of one, to remove effects of scaling and translation.

For each experiment we perform the leave-one-out evaluation of the Nearest Neighbor classification algorithm. We compared four rotation invariant shape matching experiments: 1) *ID_DTW*: Iterative deepening using DTW, 2) *ID_ED*: Iterative deepening using Euclidean distance, 3) *DTW*: using DTW, and 4) *ED*: using Euclidean distance. The results are summarized in Table 1. Euclidean distance requires no parameter, while DTW requires r, or reach of the warping window, to be defined. For our experiment, we use r=2. (Observations have shown that utilizing a Sakoe-Chiba Band with a 10% width for global constraints produces competitive results [14].)

Table 1: Summary of the averaged time (minutes) per query to the database. Accuracies are same for all four cases.

	# of elements	Accuracy ED/DTW %	ED	ID_ED	DTW	ID_DTW
Data1	1125	86.67/89.2	0.329	0.068	192.18	0.293
Data2	446	80.0/80.0	0.092	0.031	72.78	0.267

The hardware used for all experiments was a Pentium IV 2.9GHz with 2G RAM. We used Matlab to implement all algorithms. Since Matlab is an interpreted language, the overall time is slow, but we are only interested in the relative performances. Once the minimum distance of the first two shapes is calculated, initializing the *best_so_far* value, it acts as a catalyst for our iterative deepening algorithm, along with other optimizations described in sections 3.2, 3.3, and 4.3; 70% of the computational time is actually spent on the initial minimum distance calculation. As observed from Table 1, as the size of the dataset doubles, the speedup increases dramatically. The time required for searching is linear in the number of objects in the database. In contrast, with our iterative deepening methods, the bound for filtering out poor matches becomes tighter as more sequences have been observed. Because of this property, the speedup over the non-iterative algorithm is proportional to the size of the data. In other words, the larger the database of sequences, the greater the speedup obtained by our method.

6. Conclusions and future work

In this paper, we introduced a framework for rotation invariant shape matching that exploits the idea of iterative deepening along with dimensionality reduction to produce a dramatic speedup, where that gain increases with the size of database. Our algorithm has the desirable properties of providing speedup to other competitive algorithm and guaranteeing no false dismissal. Future work includes a more detailed analysis of our approach and extensions to other similarity search problems that feature distance measures that are expensive, but can be approximated at different levels of precision.

7. References

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

[2] S Antani, D. Lee, L. Long, G. Thoma, *Evaluation of shape similarity measurement methods for spine x-ray images*, J. of Vis Comm and Image Rep, 15(3): 285-302, 2004.

[3] W. Zhang, S. Dickinson, J. Feldman, S. Dunn and S. Sclaroff, *Shape-based indexing in a medical image database*, IEEE Workshop on Biomedical Image Anal., pp 221-230, 1998.

[4] The ADIAC project, http://rbg-web2.rbge.org.uk/ADIAC/

[5] A. Cardone, S. K. Gupta, 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.

[6] S. Loncaric, *A survey of shape analysis techniques*, Pattern Recognition, 31(8):983-1001, 1998.

[7] S. Chu, E. Keogh, D. Hart, M. Pazzani, *Iterative Deepening Dynamic Time Warping for Time Series*, SDM 2002.

[8] Y. Sakurai, M. Yoshikawa, C. Faloutsos: FTW: Fast Similarity Search under the Time Warping Distance. PODS 2005

[9] S. Salvador, P. Chan, *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space*, KDD Workshop on Mining Temporal and Sequential Data, pp. 70-80, 2004.

[10] N. Adams, D. Marquez, G. Wakefield, *Iterative deepening for melody alignment and retrieval*, ISMIR 2005.

[11] M. Agarwal, S. Sarangi, *Speech Recognition using Iterative Deepening Dynamic Time Warping*, PRASTUTI 2004

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

[13] J. B. Kruskall, M. Liberman, *The symmetric time warping algorithm: From continuous to discrete*, In Time Warps. Addison-Wesley, 1983.

[14] H. Sakoe, S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition*, IEEE Trans. ASSP-26(1):43- 49, Feb. 1978.

[15] F. Itakura, *Minimum prediction residual principle applied to speech recognition*, IEEE Trans. ASSP-23, 67-72, 1975.

[16] E. Keogh, *Exact Indexing of Dynamic Time Warping*, VLDB 2002: 406-417.

[17] X. Ding, W. Kong, C. Hu, S. Ma, *Image Retrieval Using Schwarz Representation of One-Dimensional Feature*, VISUAL 1999: 443-450

[18] N. Arica, F. Yarman-Vural, *Shape Similarity Measurement for Boundary Based Features*, ICIAR 2005: 431-438

[19] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, *Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases*, Knowl. Inf. Syst. 3(3): 263-286, 2000.

[20] B-K Yi, C. Faloutsos, *Fast Time Sequence Indexing for Arbitrary Lp Norms*, VLDB 2000: 385-394

[21] E. Keogh, T. Folias, The UCR Time Series Data Mining Archive, http://www.cs.ucr.edu/~eamonn/TSDMA/index.html