A Novel Technique for Indexing Video Surveillance Data

Eamonn Keogh Bhrigu Celly Chotirat Ann Ratanamahatana Victor Zordan

University of California - Riverside
Computer Science & Engineering Department
Riverside, CA 92521, USA
{eamonn, bcelly, ratana, vbz }@cs.ucr.edu

Abstract

Recent worldwide events have renewed interest in the use of video surveillance as a tool for private security, law enforcement and military applications. After appropriate feature extraction has taken place, most video surveillance problems are reduced to the problem of efficiently and robustly matching motion streams. Since all natural motion typically has some variability in the time axis, Dynamic Time Warping (DTW), a technique that aligns the motion streams before calculating their similarity, is typically used. However, DTW can only address the problem of local *scaling. As we demonstrate in this work,* uniform *scaling may be just as important for meaningful automatic analysis of video surveillance data streams. In this work, we demonstrate a novel technique to index of similarity search under uniform scaling. As we will demonstrate, our technique is simple and intuitive, and can achieve a speedup of 2 to 3 orders of magnitude under realistic settings.*

1. Introduction

Recent worldwide events have renewed interest in the use of video surveillance as a tool for private security, law enforcement and military applications. After appropriate feature extraction has taken place, most video surveillance problems are reduced to the problem of efficiently and robustly matching motion streams (i.e. time series).

The motivation for using this simple representation dates back to the classic experiments by Johansson [13]. Johansson created films of humans that are complexly blank except for a dozen or so light-points attached to actors body. He discovered that even when presented with such poor input information, people could instantly identify the actions being performed. Later studies showed simply by viewing these light points, people can even identify the gender [20] and the emotional state [25] of the actor. An additional pragmatic motivation for using a time series representation of video is the relative ease with which we can store, transmit and index time series.

Since all natural motion typically has some variability in the time axis, Dynamic Time Warping (DTW), a technique that aligns the motion streams before calculating their similarity, is typically used [1]. However, DTW can only address the problem of *local* scaling. As we demonstrate in this work, global or *uniform* scaling may be just as important for meaningful automatic analysis of video surveillance data streams. For clarity, we illustrate the difference between local scaling and uniform scaling in Figure 1.

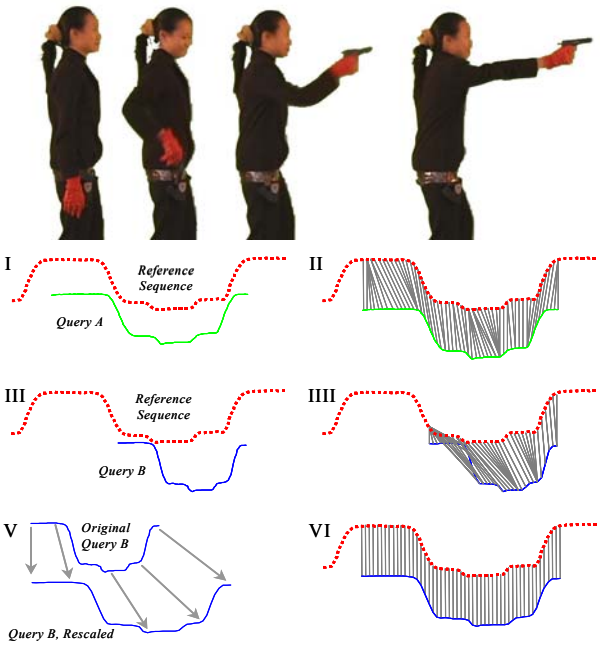

Figure 1. (*Top*) Stills from the video surveillance *gun* problem, the right hand is tracked, and converted into motion streams. (*Bottom*) Some snippets from the Y-axis of the *gun* dataset.
I) We can attempt to match the Query *A* to the most similar subsection of a longer reference dataset. II) Dynamic time warping allows small, non-linear rescaling in the Y-axis to achieve a better fit. III) Query B corresponds to a much quicker draw of the gun, and DTW (IIII) is of little utility. By simply uniformly rescaling the query to make it 34% longer (V), its true similarity (VI) to a subsection of the reference sequence is revealed.

***IWVS'03*, November 7, 2003, Berkeley, California, USA.**

The problem of efficiently indexing patterns in large time series databases has attracted great interest in the database [7][11]][[17], data mining [3][14][29], and information retrieval communities [27]. However, the importance of dealing with both uniform and local scaling does not appear to have been fully understood. Typically, we find that researchers that are interested in performance issues use the simple Euclidean distance [3][7][11][14]. However, researchers that are application driven usually understand that Euclidean distance is inappropriate for most real world applications [1][10][12][28][32]. The limitations of Euclidean distance stems from the fact that it is very sensitive to distortions in the time axis. A partial solution to this problem, Dynamic Time Warping (DTW), essentially aligns the time axis before calculating the Euclidean distance.

While dynamic time warping can only address the problem of *local* scaling, *uniform* scaling may be just as important in many domains, especially in biological and biometric domains. For example, a recent paper by Hu and Dannenberg [12] on the *query-by-humming* music retrieval problem, noted the need to "*search the database with numerous time scaled versions of queries to cover a reasonable range of queries*". While this work makes a strong empirical argument for the necessity of uniform scaling, it does not advance any technique to help mitigate its time complexity.

There exists a handful of techniques that can support similarity search under uniform scaling if the scaling factor is known in advance [4][14]; however, in most domains, it is unlikely that we know the scaling factor. In such instances, we must, as Hu and Dannenberg suggest, resort to multiple queries, one for each possible scaling factor. Clearly, this is untenable for a real time system. What we really need is a technique that can perform a single efficient query to retrieve all qualifying time series with *any* scaling. This is exactly the contribution of this paper.

The rest of this paper is organized as follows. Section 2 carefully motivates the need for similarity search under uniform scaling, and reviews related work. In Section 3 we introduce our approach to the problem. Section 4 contains an extensive empirical evaluation on 5 real world datasets. Finally, Section 5 contains conclusions and directions for future work.

2. Motivating the Need for Uniform Scaling

Many non-Euclidean distance measures for time series have been introduced; however, a recent empirical study suggests that most of them are of questionable utility [16]. Therefore, before introducing our techniques for dealing with uniform scaling, we will conduct some simple experiments to motivate its absolute need in a video surveillance setting. We begin by briefly discussing our experimental setup.

2.1 Experimental Setup

For our experiments we used a Canon ZR40 camcorder with the shutter at 1/60, video size of 720x480 pixels, and captured video at 30 frames per second. We use the technique of tracking color in the video sequence to estimate the position of objects of interest.

The position detection for the hand movement in the experiments uses a color-tracking algorithm. A frame that has good color visibility is selected from the video sequence, which in the experiment is a frame where the object of interest (in our case, the right hand) is at rest. The selected frame is then used to calculate Hue (H), Saturation (S), and Value (V) from each pixel. A region of the color to be tracked (red in this case) is also selected and this forms the region of interest (ROI). The HSV from each pixel of the selected frame, along with the mean and covariance from the ROI of the selected frame form the input to find the probability distribution for the ROI over the whole image [8]. The probability distribution uses a multivariate Gaussian and results in a probability matrix the size of the image. This resultant matrix is converted to a binary image by thresholding, and then the resultant binary image is used to compute the centroid position of the hand. We note that our feature extraction system is somewhat naïve, since the main contribution of this work is indexing of the extracted data. More sophisticated and realistic techniques for human body motion acquisition from video can be found in the literature [5][15][21][22].

2.2 The Gun Problem Experiment

We recorded a dataset to use as reference containing two classes, each with 50 examples. All instances were created using the same female actor (who is 155 cm tall), in a single session. The two classes are:

- **Gun-Draw**: The actor has her hands by her side. She draws a replicate gun from a hip mounted holster, points it at a target for approximately one second, then returns the gun to holster, and her hands to her side.
- **Point**: The actor has her hands by her side. She points with her index finger to a target for approximately one second, and then returns her hands to her side.

For both classes, we tracked the centroid of the right hand in both the X- and Y-axis of the image plane; however, in the experiments that follow, we will consider just the Y-axis for simplicity.

The overall motions of both classes are very similar. However, it is possible for humans to visually classify the two classes with great accuracy, after noting that the actor must lift her hand above holster, then reach down for the gun, this action creates a subtle distinction between the classes as seen in Figure 2.

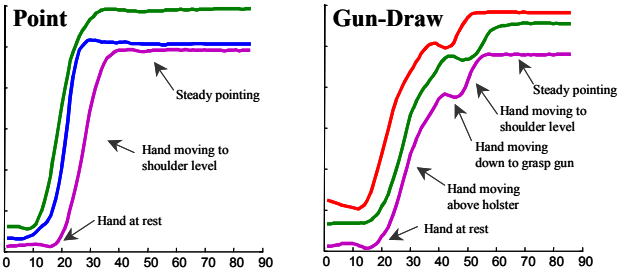

Figure 2. Sample Y-axis motion streams from the video surveillance *gun* problem. (*Left*) Samples of the **point** class are characterized by a smooth transition from the resting position to the pointing position. (*Right*) Samples of the **Gun-Draw** class are superficially similar, but are characterized by a dip where the actor reaches down towards the holster to grasp the gun.

We created a similar dataset with another actor, this time a male, who is 190cm tall. The difference in height between the two actors is not too much of a problem, since a standard preprocessing step in matching motion streams is to perform Z-normalization before performing comparisons [3][19]. However, detailed visual inspection of both the video and the extracted time series suggests that the male actor draws his gun more slowly.

We performed the following experiment; we manually extracted the 50 male-Gun-Draw sequences beginning with just before the gun is drawn, and ending approximation 0.5 seconds after the gun was pointed at the target. We used these sequences as a query to the female reference dataset, recording a match to the Gun-Draw class as a success, and a match to the Point class as a failure. We compare two distance measures, the classic Euclidean distance [3][7][17][18] and the Euclidean distance after allowing any uniform scaling between 70% and 130%. Table 1 summarizes the results:

Table 1. A comparison of accuracy and time for two approaches on the Gun problem.

	Accuracy	Time (sec)
Euclidean Distance	54%	0.89
Uniform Scaling	96%	51.4

The results for Euclidean distance are little better than random guessing; on the other hand, uniform scaling works quite well. The intuition for its success is as follows; without rescaling, the two time series only match in the coarse details (they both begin with low Y values, and transition to high Y values, where they hold steady). However, after uniform scaling corrects for the differences in speed of the two actors, the smaller details (i.e. the "dip" in the Gun-Draw class) that actually differentiate the classes, become relatively more important.

These results strongly reiterate the utility of uniform scaling, they also show that, naively calculated, it may be too slow for practical real time systems.

2.3 Related Work

In the past decade, there have been literally hundreds of papers on similarity search using the Euclidean distance [3][7][17][18]; useful surveys can be found in [11] and [29]. However, there has been an increasing awareness that the Euclidean distance may be unsuitable for many applications [1][16][31][32], particularly for applications that involve biological processes such as gait [10], gene expression [6] and video surveillance [2].

Dozens of non Euclidean distance measures for time series have been introduced in the last decade, however, a recent empirical study suggests that most of them are of debatable utility [16]. The only non-Euclidean distance measure that has been forcefully shown to be superior to Euclidean distance is DTW; its utility has been demonstrated in domains as diverse as bioinformatics [1], chemical engineering, gait analysis [10], speech recognition, meteorology, music retrieval [12][32], and robotics. However, DTW only considers local stretching and shrinking of the time axis. As we demonstrated in the previous section, uniform scaling may be equally important for video surveillance.

The utility of uniform scaling has been noted before [12][14][26][27]. However, all previous work has focused on speeding up similarity search, when the scaling factor is *known*. For example, there are systems that can index data of length 200, and support queries of any length from 150 to 200. However, the user must specify what query length they wish to run, perhaps a query of length 175. If the user wishes to find the best matching time series, at any length from 150 to 200, they would have to run every possible query, of length 150, 151 ,…, 200 to find the answer. This is clearly untenable. As all these systems claim about one order of magnitude speed up, placing them in a loop and running them 50 times is clearly going to be self defeating. The feature that differentiates our work from the rest is that we allow a user to issue a single query, and find the best match at *any* scaling. Our proposed technique is unique in this aspect.

Since the main contribution of this work is in feature indexing, not feature extraction, we have not considered related work in the vast body of literature on capturing the human motion from video (and other sources), instead

we refer the interested reader to [24] which contains an excellent overview.

3. Uniform Scaling

We begin by formally defining the uniform scaling problem. Although our data type of interest is variously called "*motion streams*", "*trajectories*", "*sequences*", etc, we will hereafter use the unifying term "*time series*". Although in the video surveillance domain, we may be ultimately interested in working with two or three-dimensional data [6][15][21][22][23][31], we will consider only one-dimensional time series for simplicity.

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

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

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

For clarity of presentation, we will assume that $n \le m$, that is to say, C is always longer than or equal to Q, and thus we are only interested in stretching the query to match some prefix of C. This assumption is only to simplify notion and does not preclude matching a time series by shrinking, since we can always reverse the roles of the sequences.

If we wish to compare the two time series, and it happens that $n = m$, we can use the ubiquitous Euclidean distance:

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \quad (3)$$

Since the square root function is monotonic and concave, we can remove the square root step and get identical rankings, clustering and classifications. This measure is called the squared Euclidean distance:

$$D(Q,C) \equiv \sum_{i=1}^{n} (q_i - c_i)^2 \quad (4)$$

In addition to the utility of slightly speeding up the calculations, working with this distance measure makes other optimizations possible [19].

If n is smaller than m, then the distance measures introduced above are not defined. To compare the two time series in this case, we have several choices; we can truncate C and compare Q to $[c_1, c_2, \ldots, c_n]$, or we can somehow *stretch* Q to be of length m, or more generally we can *stretch* Q to be of length p, ($n \le p \le m$), truncate off the last m-p values of Q, then use squared Euclidean distance. The informal idea behind *stretching* can be captured in the more formal definition of scaling. To scale time series Q to produce a new time series QP of length p, the formula is:

$$QP_j = Q_{\lceil j * n/p \rceil}, \ 1 \le j \le p \quad (5)$$

Note that we can quickly obtain any scaling in O(p) time. We call the ratio p/n the *scaling factor* or *sf*. Slightly different definitions of scaling do exist, but they do not affect the results that follow. Figure 3 visually summarizes the above definitions.

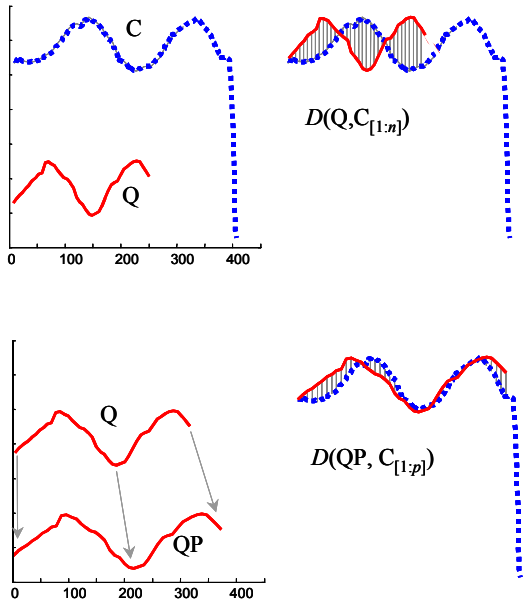

Figure 3. A visual summary of the notation introduced in this section. (*Top-left*) A candidate time series C, and a shorter query Q. (*Top-right*) The squared Euclidean distance between Q and the first n data points in C can be visualized as the sum of the squared lengths of the gray hatch lines. (*Bottom-left*) The query Q can be stretched to length p, producing a new time series QP. (*Bottom-right*) In this case, QP is a good match to the first p data points in C.

3.1 Brute Force Search Under Uniform Scaling

If we wish to find the best scaled match between Q and C, we can simply test all possible scalings, as illustrated in Table 2.

Table 2. An algorithm to find the best scaled match between two time series

```
Algorithm:
Test_All_Scalings(Q,C)
best_match_val        = inf;
best_scaling_factor   = null;
for p = n to m
 QP = rescale(Q,p);
 distance = squared_Euclidean_dist(QP, C[1..p]);
 if distance <  best_match_val
              best_match_val = distance;
              best_scaling_factor = p/n;
 end;
end;
return(best_match_val, best_scaling_factor)
```

The algorithm takes only O(p*(m-n)) time and seems unworthy of any optimization effort. However, when indexing real world datasets, rather than having a single candidate time series C, we are typically confronted with massive collection of possible candidate time series, which will denote as **C**. To find the best scaled match to a query Q, in data collection **C**, we can use a brute force algorithm as shown in Table 3.

Note that the time complexity for this algorithm is O(|**C**| * (m-n)), which is simply untenable for large datasets.

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

```
Algorithm:
Search_Database_for_Scaled_Match(Q,C)
overall_best_time_series = null;
overall_best_match_val   = inf;
overall_best_scaling     = null;
for i = 1 to number_of_time_series_in_(C)
  [dist, scale] = Test_All_Scalings(Q,Ci)
  if dist <  overall_best_match_val
       overall_best_time_series = i;
       overall_best_match_val   = dist;
       overall_best_scaling     = scale;
  end;
end;
return(overall_best_time_series,
        overall_best_match_val, overall_best_scaling)
```

3.2 Speeding up Search with Lower Bounding

To speed up matching under uniform scaling we will rely on the classic idea of lower bounding. The intuition is this: given some technique for quickly calculating the minimum possible distance between the query and a candidate sequence at any possible scaling, we can prune off many calculations. In more detail, we maintain a variable that contains the distance of the best-scaled match encountered thus far. Before calling the subroutine `Test_All_Scalings` on the next candidate time series, we first perform the quick lower bounding test. If the lower bound distance between the candidate and the query is greater than the distance of the best-scaled match already seen, we can simply discard the candidate from consideration. For clarity, the idea is formalized in Table 4, although the algorithm differs from the algorithm in Table 3 only in the addition of the lower bounding test as a precondition to the subroutine `Test_All_Scalings`.

Table 4. A modified algorithm for searching for the best match under uniform scaling

```
Algorithm:
Faster_Search_Database_for_Scaled_Match(Q,C)
overall_best_time_series = null;
overall_best_val         = inf;
overall_best_scaling     = null;
for i = 1 to number_of_time_series_in_(C)
 if lower_bound_distance(Q,Ci) <overall_best_val
 [dist, scale] = Test_All_Scalings(Q,Ci)
  if dist <  overall_best_match_val
       overall_best_time_series = i;
       overall_best_match_val   = dist;
       overall_best_scaling     = scale;
  end;
 end;
end;
return(overall_best_time_series,
        overall_best_match_val, overall_best_scaling)
```

There are only two important properties of a lower bounding measure:

- It must be fast to compute. A measure that takes as long to compute as `Test_All_Scalings` is of little use. We would like the time complexity to be at most linear in the length of the time series.
- It must be a relatively tight lower bound. A function can achieve a trivial lower bound by always returning zero as the lower bound estimate. However, in order for the algorithm in Table 4 to be effective, we require a method that tightly bounds the value of the best match.

The idea of speeding up search using lower bounding is not new; in fact, it is the cornerstone of virtually every similarity search algorithm. However, while dozens of lower bounding measures are known for Euclidean distance [3][7][14][17][18], and three lower bounding measures known for DTW [16][32], there is no lower bounding measure in the literature for uniform scaling. In the next section, we introduce the first such measure.

3.3 Lower Bounding Uniform Scaling

To create a lower bounding distance measure for uniform scaling we will generate a bounding envelope. Bounding envelopes were introduced in [16] to lower bound DTW, and since then they have sparked a flurry of research activity [9][28][31][32]. While the principle is the same here, the definitions of the envelope are very different. In particular, we create two sequences U and L, such that:

$$U_i = \max(c_{\lfloor (i-1)*m/n \rfloor +1}, \ldots, c_{\lfloor i*m/n \rfloor}) \quad (6)$$

$$L_i = \min(c_{\lfloor (i-1)*m/n \rfloor +1}, \ldots, c_{\lfloor i*m/n \rfloor}) \quad (7)$$

These sequences can be visualized as bounding the first n points of the time series C. Figure 4 shows some examples.

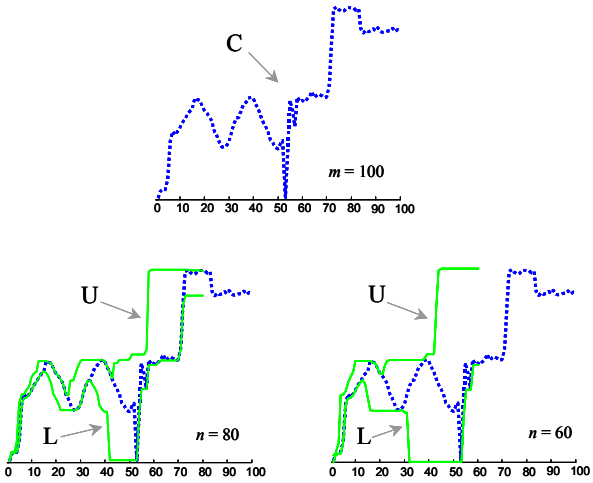

Figure 4. (*Top-center*) A time series C of length 100. (*Bottom-left*) The time series shrouded by upper and lower envelopes U and L with lengths 80. (*Bottom-right*) The same time series shrouded by upper and lower envelopes U and L with lengths 60.

Having defined the U and L, we can now discuss the lower bounding function; it is a modification of the one introduced in [16] for the problem of DTW.

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

This function can be visualized as the squared Euclidean distance between any part of the query time series not falling within the envelope and the nearest (orthogonal) corresponding section of the envelope. Figure 5 illustrates the idea.

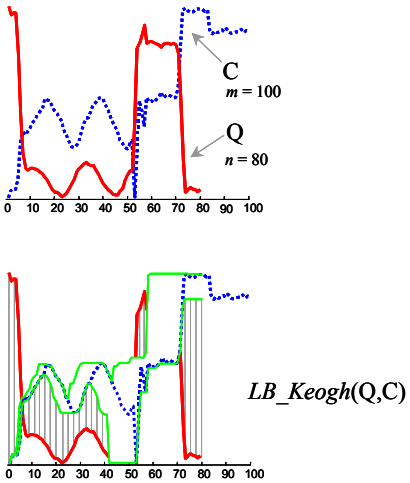

Figure. 5. (*Top*) A time series C and a shorter query Q. (*Bottom*) A visualization of the lower-bounding function *LB_Keogh*(Q,C). Note that any part of query time series Q that falls inside the bounding envelope is ignored. Otherwise, the distance corresponds to the sum of the squared straight-line distances from the query to the nearest point in the envelope (the gray hatch lines).

We have claimed that *LB_Keogh*(Q,C) lower bounds the squared Euclidean distance between any scaling of Q, and the appropriate prefix of C. The proof is straightforward; we omit it for brevity.

3.4 Indexing under Uniform Scaling

As noted in Section 3.2, if we have a distance measure that is expensive in terms of CPU time, we can dramatically speed up similarity search using a tight lower bound. However, if the majority of the data exists on secondary storage, the CPU costs may be dwarfed by the disk (or tape) access time. The solution is to *index* the data.

Fortunately, the ability to index uniform scaling essentially comes for free! A technique for indexing envelopes under *LB_Keogh* was introduced in [16]. Since then, many other researchers have used this technique and suggested extensions [9][28][31][32]. This explosion of interest has ensured that indexing of time series envelopes has become a mature technology in only one year. For completeness, we will discuss the minor extensions to [16] necessary to index under uniform scaling.

We have previously denoted a time series as $C = c_1, \ldots, c_n$. We assume each sequence in our database is n units long. Let N be the dimensionality of the space we wish to index ($1 \le N \le n$). For convenience, we assume that N is a factor of n.

A time series C of length n can be represented in N dimensional space by a vector $\overline{C} = \overline{c}_1, \ldots, \overline{c}_N$. The i^{th} element of $\overline{C}$ is calculated by the following equation:

$$\overline{c}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} c_j \quad \textbf{(9)}$$

In other words, to reduce the time series from n dimensions to N dimensions, the data is divided into N equal sized "frames". The mean value of the data falling within a frame is calculated and a vector of these values becomes the data reduced representation. The representation can best be visualized as an attempt to model the original time series with a linear combination of box basis functions as shown in Figure 6.

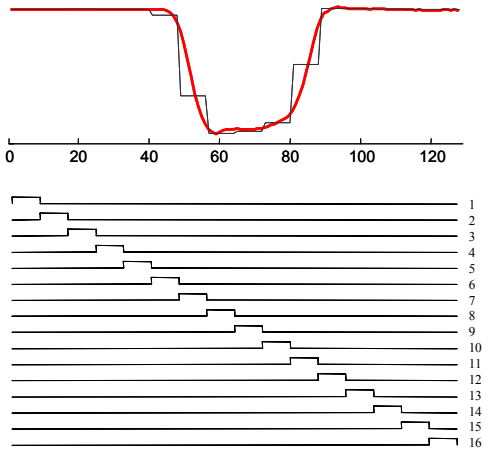

Figure 6. The PAA representation can be visualized as an attempt to model a sequence with a linear combination of box basis functions. In this example, the first 128 data points of one of the *gun-problem* instances are reduced to 16 dimensions.

3.4.1 Modifying PAA to index uniform scaling queries

In Section 3.3, we discussed the lowering bounding function *LB_Keogh*. However, calculating this function requires *n* values. Since *n* may be in the order of hundreds to thousands for realistic human motion, and multi-dimensional index structures begin to degrade rapidly somewhere above 16 dimensions, we need a way to create a lower, *N* dimension version of the function, where *N* is a number that can be reasonably handled by a multi-dimensional index structure. We also need this lower dimension version of the function to lower bound *LB_Keogh* (and therefore, by transitivity, uniform scaling).

We begin by creating special piecewise constant approximations of U and L, which we will denote as $\hat{U}$ and $\hat{L}$. Although they are piecewise constant approximations, the definitions of $\hat{U}$ and $\hat{L}$ differ from those we have seen in Eq. 6 and 7. In particular, we have

$$\hat{U}_i = \max\left(U_{\frac{n}{N}(i-1)+1}, \ldots, U_{\frac{n}{N}(i)}\right) \quad \textbf{(10)}$$

$$\hat{L}_i = \min\left(L_{\frac{n}{N}(i-1)+1}, \ldots, L_{\frac{n}{N}(i)}\right) \quad \textbf{(11)}$$

We can visualize $\hat{U}$ and $\hat{L}$ as the piecewise constant functions which bound, without intersecting, U and L, respectively. Figure 7 illustrates this intuition.

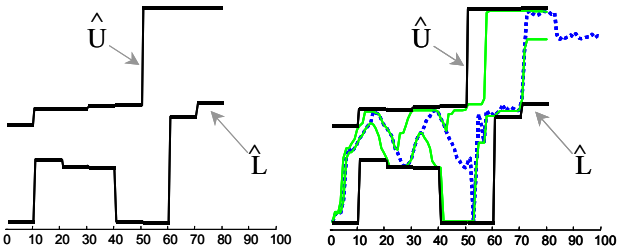

Figure 7. We can readily visualize of $\hat{U}$ and $\hat{L}$ as the piecewise constant functions which bound, without intersecting, U and L, respectively. (*Left*) The $\hat{U}$ and $\hat{L}$ for the time series shown in Figure 4. (*Right*) The $\hat{U}$ and $\hat{L}$ shown overlaid on top of the generating time series.

We are now able to define the low dimension, lower bounding function, which we denote as LB_PAA. Given a candidate sequence C, transformed to $\overline{C}$ by Eq. 9, and a query sequence Q, with its companion PAA functions $\hat{U}$ and $\hat{L}$, the following function lower bounds *LB_Keogh*

$$LB\_PAA(Q,\overline{C}) = \sqrt{\sum_{i=1}^{N} \frac{n}{N} \begin{cases} (\overline{c}_i - \hat{U}_i)^2 & \text{if } \overline{c}_i > \hat{U}_i \\ (\overline{c}_i - \hat{L}_i)^2 & \text{if } \overline{c}_i < \hat{L}_i \\ 0 & \text{otherwise} \end{cases}} \quad \textbf{(12)}$$

The final step necessary to allow indexing is to define a MINDIST(Q,R) function that returns a lower bounding measure of the distance between a query Q and R, where R is its Minimum Bounding Rectangle (MBR).

Suppose our index structure contains a leaf node **U**. Let R = (L, H) be the MBR associated with **U** where L = $\{l_1, l_2, \ldots, l_N\}$ and H = $\{h_1, h_2, \ldots, h_N\}$ are the lower and higher endpoints of the major diagonal of R. By definition, R is the smallest rectangle that spatially contains each PAA point $\overline{C} = \overline{c}_1, \ldots, \overline{c}_N$ stored in **U**. Given the above, MINDIST(Q,R) is defined as:

$$MINDIST(Q,R) = \sqrt{\sum_{i=1}^{N} \frac{n}{N} \begin{cases} (l_i - \hat{U}_i)^2 & \text{if } l_i > \hat{U}_i \\ (h_i - \hat{L}_i)^2 & \text{if } h_i < \hat{L}_i \\ 0 & \text{otherwise} \end{cases}} \quad \textbf{(13)}$$

This function is visualized in Figure 8.

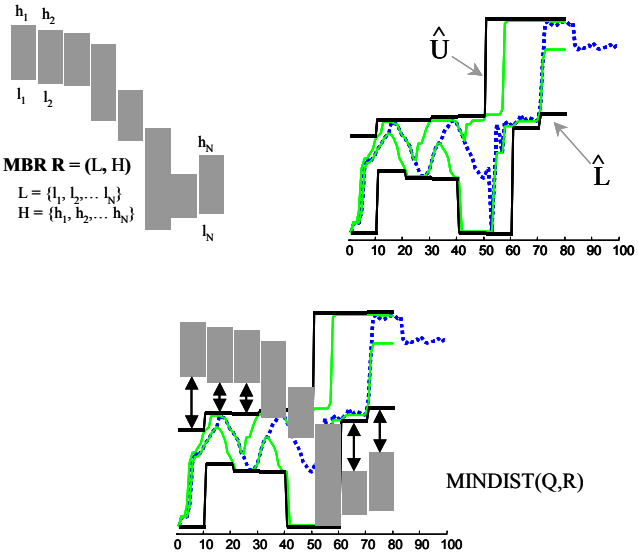

Figure 8. A) A representation of a Minimum Bounding rectangle (MBR). **B**) A subsection of the query shown in Figure 4, with its attendant functions $\hat{U}$ and $\hat{L}$. **C**) An illustration of the MINDIST function. The lengths of the arrow lines, squared, scaled by *n*/*N*, summed and square rooted, are returned as the minimum distance between Q and any sequence contained within R

Having defined a lower bounding function between a query time series and all scalings (*LB_Keogh*), and an a lower bound between a query and all objects in a MBR (MINDIST) we can use any of the many K-Nearest Neighbor search algorithms in the literature. In this work, we use the algorithm proposed in [30]. The basic intuition of the algorithm is to use a priority queue to prioritize the order in which to test the full Euclidean distance (at arbitrary scales). Two types of things can be inserted into the priority queue, MBRs and {U, L} tuples, sorted by MINDIST and *LB_Keogh,* respectively. As soon as the *best-so-far* distance is less than the value at the head of the queue, the search is finished. Until that point is reached, the object at the head of the queue is dequeued. If it is an MBR, the corresponding {U, L} tuples are enqueued; if, on the other hand, it is a {U, L} tuple, algorithm `Test_All_Scalings(Q,C)` is performed on the corresponding candidate time series and the *best-so-far* time series distance is updated as appropriate. We encourage the reader to consult [30] for more details.

4. Experimental Results

In this section, we test our proposed approach with a comprehensive set of experiments. We compare only to the brute force search algorithm defined in Table 3, because there are no other techniques that support uniform scaling queries with a single query. To eliminate the possibility of implementation bias [19], we will report the *Pruning Power*, the fraction of times that our approach must call the squared Euclidean distance function.

$$Pruning\ Power = \frac{Number\ of\ calls\ to\ distance\ function\ by\ proposed\ approach}{Number\ of\ calls\ to\ distance\ function\ by\ brute\ force\ search} \quad \textbf{(14)}$$

This measure depends only on the tightness of the lower bounds, and is independent of language, platform, caching or any other implementation details. As an additional sanity check, we also measured the CPU time. However, since it is almost perfectly correlated with the *Pruning Power*, we will omit it for brevity.

It has been forcefully demonstrated that the quality of lower bounding measures, and therefore the speed of search, can vary greatly depending on the data [19]. We therefore tested our approach on a variety of datasets. Since we currently have only one video surveillance dataset, we also used other biomedical and biomechanical datasets from the literature. Figure 9 shows a sample of each.

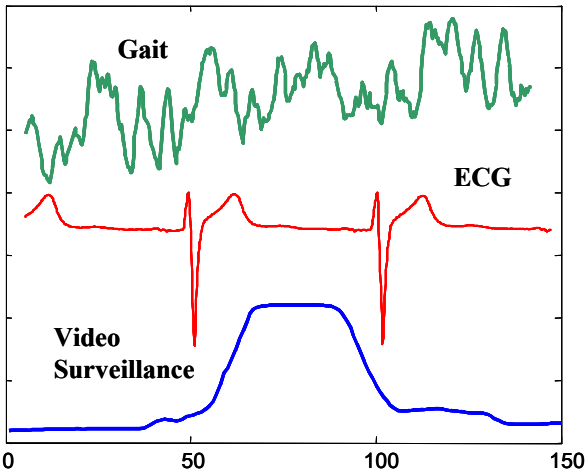

Figure 9. Randomly extracted samples of the time series datasets used in the experiments. The gait dataset came from PhysioBank, and the ECG dataset came from the UCR time series archive.

Since the speedup obtained from our approach clearly depends on range of scaling factors and the length of the time series, we will test our approach for the cross product of scaling factors = {1.05, 1.10, 1.15, 1.20, 1.25} and time series candidate lengths of {16, 32, 64, 128, 256}.

We conducted our experiments as follows. We randomly removed a subsequence of the appropriate length from the data to use as a query, and then we randomly chose 5,000 other subsequences to act as the database. We then searched for the best scaled match, noting the pruning power. We repeated this 100 times for every combination of scaling factors and candidate lengths. Figure 10. shows the results.

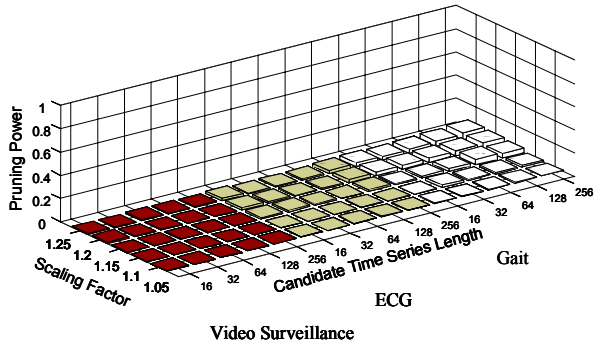

Figure 10. The pruning power of *LB_Keogh* of 3 different datasets, over a range of scaling factors and candidate lengths

The results are quite impressive. The worst case is a single order of magnitude speedup; more generally, two to three orders of magnitude speedup are observed. Note that the pruning power seems independent of the candidate time series lengths, but does get worse as the scaling factor increases. This is to be expected since for large scaling factors, the *LB_Keogh* function has relatively little information with which to calculate the lower bound.

As with many indexing techniques, the pruning power of our approach improves with the size of the dataset. The intuition behind this effect is that the larger the dataset, the more likely we are to find a very close match early on in the search, and thus derive the maximum benefit from the lower bound pruning test (the outermost `if` statement in Table 4). To demonstrate this, we repeated the previous experiment for different size datasets. The results for just the video surveillance dataset are shown in Figure 11.

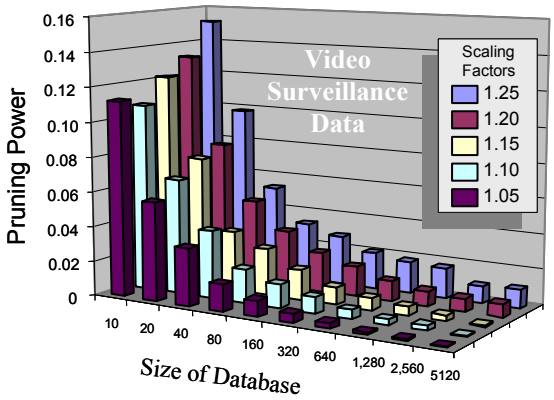

Figure. 11. The pruning power of *LB_Keogh* on the burst dataset, over a range of scaling factors and database sizes. Note the scale of the Z-axis is different from that of Figure 10.

The results clearly show that as the database size increases, the pruning power improves. This is a very desirable property when indexing larger datasets.

5. Conclusions

We have shown how to index similarity search under uniform warping. We motivated the need for such an ability by showing that for at least some video surveillance problems, small changes in scale can completely confuse the other similarity measures.
We plan to extend this work in several directions. First we hope to extend the matching under uniform scaling to multi-dimension time series, second, we plan to incorporate both DTW and uniform scaling into a single unified framework.

Reproducible Results Statement: All datasets and code used in this paper are available for free, by emailing the authors.

6. References

[1] Aach, J. and Church, G. (2001). Aligning gene expression time series with time warping algorithms. Bioinformatics. Volume 17, pp 495-508

[2] Alon, J., Sclaroff, S., Kollios, G., and Pavlovic, V. (2003). Discovering Clusters in Motion Time-Series Data. In the Proc. of the IEEE CVPR 2003

[3] Chan, K. & Fu, A. W. (1999). Efficient time series matching by wavelets. In *proceedings of the 15th IEEE Int'l Conference on Data Engineering*. Sydney, Australia. pp 126-133

[4] Chu, K., Lam., S. & Wong, M. (1998) An Efficient Hash-Based Algorithm for Sequence Data Searching. The Computer Journal 41 (6): 402-415

[5] Comaniciu, D., Ramesh, V., and Peter Meer (2003). Kernel-Based Object Tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(5): 564-575 (2003)

[6] Corradini. A. (2001). Dynamic Time Warping for Off-line Recognition of a Small Gesture Vocabulary. In *Proceedings of the 2nd IEEE International Conference on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, pp. 82-89.

[7] Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *In Proc. ACM SIGMOD Conf.*, Minneapolis. pp. 419-429

[8] Forsyth, D.A. and Ponce, J. (2001). Computer Vision: a modern approach, Prentice-Hall, 2001

[9] Fung, W and Wong, M. (2003). Efficient Subsequence Matching for Sequences Databases under Time Warping . The 7th International Database Engineering and Application Symposium Hong Kong.

[10] Gavrila, D. M. & Davis,L. S.(1995). Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In *International Workshop on Automatic Face- and Gesture-Recognition*

[11] Hetland, M. (2003). A Survey of Recent Methods for Efficient Retrieval of Similar Time Sequences. To appear in an Edited Volume*, Data Mining in Time Series Databases*. Published by the World Scientific Publishing Company

[12] Hu, N. and Dannenberg, R.B. (2002). A comparison of melodic database retrieval techniques using sung queries. ACM/IEEE Joint Conference on Digital Libraries, JCDL 2002, Portland, Oregon, USA, June 14-18, 2002: pp. 301-307

[13] Johansson, G. (1973). Visual perception of biological motion and a model for its analysis. Perception and Psychophysics, 14:210-211, 1973.

[14] Kahveci, T. & Singh, A. (2001). Variable length queries for time series data. In *proceedings of the 17th Int'l Conference on Data Engineering*. Heidelberg, Germany, pp 273-282

[15] Kakadiaris, I.A. and Metaxas, D. (2000). Model-based estimation of 3D human motion. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(12), pp. 1453-1459, 2000

[16] Keogh, E. (2002). Exact indexing of dynamic time warping. In 28th *International Conference on Very Large Data Bases*. Hong Kong. pp 406-417

[17] Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra (2000). Dimensionality reduction for fast similarity search in large time series databases. Journal of Knowledge and Information Systems. pp 263-286

[18] Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra (2001) Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc of ACM SIGMOD Conference on Management of Data*. pp 151-162

[19] Keogh, E. and Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In *the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Canada. pp 102-111.

[20] Kozlowski L T & Cutting J E (1977) Recognizing the gender of walkers from dynamic point-light displays. Perception and Psychophysics 21 575-580.

[21] Marchesotti, L., Marcenaro, L., and Regazzoni, C.S. (2002) Tracking and counting multiple interacting pedestrian in indoor scenes, Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance June 1, 2002 (in conjunction with ECCV'02) Copenhagen, Denmark. (2002 IEEE)

[22] Mikic, I., Trivedi, M., Hunter, E., and Cosman, P. (2002). Human Body Model Acquisition and Tracking using Voxel Data, International Journal of Computer Vision, 2002.

[23] Mittal, A. and Davis, L.S. (2003). M2Tracker: A Multi-View Approach to Segmenting and Tracking People in a Cluttered Scene. Anurag Mittal and Larry S. Davis. International Journal of Computer Vision. Vol. 51 (3), Feb/March 2003

[24] Moeslund, T. and Granum, E. (2001). A survey of computer vision-based human motion capture. CVIU, 81(3):231--268, 2001.

[25] Montepare, J. M., & Zebrowitz-McArthur, L. (1988). Impressions of people created by age-related qualities of their gaits. Journal of Personality and Social Psychology, 55, 547-556.

[26] Park, S., Chu, W. W., Yoon, J. & Hsu, C. (2000). Efficient searches for similar subsequences of different lengths in sequence databases. In *proceedings of the 16th Int'l Conference on Data Engineering*. San Diego, CA, pp 23-32

[27] Perng, C., Wang, H., Zhang, S., & Parker, S. (2000). Landmarks: a new model for similarity-based pattern querying in time series databases. In *proceedings of 16th International Conference on Data Engineering*. pp 33-42

[28] Rath, T. & Manmatha, R. (2002): Lower-Bounding of Dynamic Time Warping Distances for Multivariate Time Series. Tech Report MM-40, University of Massachusetts Amherst.

[29] Roddick, J. F. and Spiliopoulou, M. (2001). A Survey of Temporal Knowledge Discovery Paradigms and Methods. *IEEE Tran's on Knowledge and Data Engineering*. pp. 750-767

[30] Seidl, T. & Kriegel, H. (1998). Optimal multi-step k-nearest neighbor search. *SIGMOD Conference*. pp 154-165

[31] Vlachos, M., Kollios, G., & Gunopulos, G. (2002). Discovering similar multidimensional trajectories. In *Proc 18th International Conference on Data Engineering*

[32] Zhu, Y. & Shasha, D. (2003). Query by Humming: a Time Series Database Approach. SIGMOD 2003.