A Monte Carlo Approach to Biomedical Time Series Search

Jonathan Woodbridge, Bobak Mortazavi, Majid Sarrafzadeh
Computer Science Department
University of California, Los Angeles
Email: {jwoodbri, bobakm, majid}@cs.ucla.edu

Alex A.T. Bui
Medical Imaging Informatics
University of California, Los Angeles
buia@mii.ucla.edu

***Abstract*—Time series subsequence matching (or signal searching) has importance in a variety of areas in health care informatics. These areas include case-based diagnosis and treatment as well as the discovery of trends and correlations between data. Much of the traditional research in signal searching has focused on high dimensional R-NN matching. However, the results of R-NN are often small and yield minimal information gain; especially with higher dimensional data. This paper proposes a randomized Monte Carlo sampling method to broaden search criteria such that the query results are an accurate sampling of the complete result set. The proposed method is shown both theoretically and empirically to improve information gain. The number of query results are increased by several orders of magnitude over approximate exact matching schemes and fall within a Gaussian distribution. The proposed method also shows excellent performance as the majority of overhead added by sampling can be mitigated through parallelization. Experiments are run on both simulated and real-world biomedical datasets.**

***Keywords*-Subsequence Matching, Signal Searching, Biomedical Time Series**

I. Introduction

Subsequence matching is the process of finding similar segments within a database of time series signals. Subsequence matching (or signal searching) has importance in a variety of areas in health care informatics. These areas include case-based diagnosis and treatment as well as the discovery of trends and correlations between data. Much of the traditional research in signal searching has focused on high dimensional approximate exact matching (R-NN). A match is defined as any two segments $u, v \in S$ such that $dist(u, v) \leq R$ where S is the search space, $dist$ is a measure of distance (such as Euclidean distance), and R is a predefined threshold. In practice, R tends to be relatively small, leading to homogeneous result sets. While results may be precise, they offer little information gain. Of course, result sets can be enlarged by increasing R. However, arbitrarily increasing R can destabilize the result set, rendering it meaningless [1].

This paper presents a randomized Monte Carlo approach for improving search results. This approach enlarges search results while ensuring precision and yielding higher relative information gain. This method is built upon two assumptions: time series databases are extremely large [2], and result sets follow a Gaussian distribution [3][4]. The proposed method consists of two steps. First, a query segment q undergoes m randomizations constructing a set Q of query segments where $|Q| = m$. Next, an R-NN search for each $u \in Q$ is performed using the l_2 norm (Euclidean distance). The Euclidean distance between q and all segments $u \in Q$ follows a Gaussian distribution with a mean μ_Q and standard deviation σ_Q determined by the randomization.

Locality Sensitive Hashing (LSH) [5] is employed as the underlying hash-based nearest neighbor search algorithm. Under an LSH family of hash functions, similar objects have a higher probability of collision (and vice-versa). A search using LSH has a proven sub-linear computational complexity. This is unlike spatial time series databases that have been shown both theoretically and experimentally to perform worse than sequential search for data with as little as ten dimensions [6].

Results from this paper are shown both theoretically and experimentally. Experiments are run on both synthetic random walk and real-world publicly available datasets. The randomized approach increased the number of search results by several orders of magnitudes over LSH alone while keeping similar preciseness. Experimental databases contained tens of millions of indexed subsequences showing both correctness and scalability. However, the proposed algorithm is highly parallelizable, potentially allowing for databases of a much larger scale.

This paper is organized as follows: Section II presents the motivation and related work in signal searching; Section III provides the proposed method as well as its theoretical proof; and Section IV describes the experimental set-up with the results and relating discussion in Section V. Conclusions are given in Sections VI.

II. Background

Subsequence matching, or signal searching, is synonymous with the R-NN problem in high dimensional space. Nearest neighbors are defined as all objects that fall within distance R to the query object. For the purpose of this paper, distances are defined by the ℓ_2 norm (Euclidean distance) and objects are represented as points in $\Re^d$. Sequential search is a naïve approach to solving the R-NN problem. This, of course, has an $O(n)$ computational complexity

978-1-4673-2560-8/12/$31.00 ©2012 IEEE

that is far too slow for large databases. Hence, an optimal solution would guarantee sub-linear complexity.

Previous works have attempted to solve the R-NN problem by reducing feature vectors' dimensions and applying spatial indexing for search [7][8]. However, spatial indexes were shown, both theoretically and experimentally, to perform worse than sequential search for data with as little as ten dimensions [6]. Locality Sensitive Hashing (LSH) [5] was introduced as an alternative to spatial indexing schemes. Unlike spatial indexing schemes, LSH has a proven sub-linear computational complexity.

An LSH scheme guarantees, with probability p_1, that all segments within distance R to the query segment are returned. In addition, all segments that fall at a distance greater than cR (where C is a predefined constant) are not returned with probability p_2. LSH results are pruned such that all segments greater than distance R are suppressed. [5] show that the total query runtime is sub-linear and dominated by $O(n^\rho)$ distance computations (pruning) where $\rho = \frac{\ln 1/p_1}{\ln 1/p_2}$. The experimental implementation in this paper uses the LSH scheme based on p-stable distributions defined in [9].

In general, an R-NN query suffers from homogeneity as results are extremely similar to the query segment. Raising the value of R is one naïve approach to increasing heterogeneity. However, this approach would add instability to the nearest neighbor problem, rendering the solution meaningless [1]. Therefore, the expansion of search results must be bounded to ensure stability. The solution proposed by this paper is founded on the property that, in terms of classification, classes are composed of multiple sub-groupings that are Gaussian distributed [3][4].

III. Method

The following solution is founded on two assumptions:

1) Time series databases are extremely large; and
2) Result sets follow a normal distribution.

The first assumption implies that only a subset of true matches is required by a query, and therefore, an accurate sampling is sufficient. The second assumption results from the finding that, in terms of classification, classes are composed of multiple sub-groupings that are Gaussian distributed [3][4]. This assumption allows for a Monte Carlo estimation of the true result set. Distances from a search segment s to its result set S are therefore modeled as a normal distribution $N(\mu_S, \sigma_S)$. m randomizations of a query segment s are used as queries into a time series database using Locality Sensitive Hashing (LSH). The randomizations of s are created such that the query results $\hat{S}$ are a Monte Carlo approximation of S and therefore form a normal distribution with μ_S and σ_S

Each time series segment is assumed to be normally distributed ($N(0,1)$). The distance between two time series segments s and $\hat{s}$ is defined as follows:

Table I
LOOKUP TABLE FOR μ AND σ ASSUMING A $\sigma_d = 1$.

n	μ	σ^2
128	11.2916	.4990
256	15.9844	.4995
512	22.6164	.4998
1024	31.9922	.4999
2048	45.2493	.4999

$$\frac{dist(s,\hat{s})}{2} = \sqrt{\sum_{i=0}^{n}\left(\frac{s(i)-\hat{s}(i)}{2}\right)^2}, \tag{1}$$

where 2 is a normalization factor due to the fact that the distribution of the differences between two $N(0,1)$ variables is $N(0,2)$. This results in $dist(s,\hat{s})$ being distributed as a χ distribution with the following property:

$$\lim_{n\to\infty} \frac{dist(s,\hat{s})/2 - \mu_d}{\sigma_d} \sim N(0,1), \tag{2}$$

where μ_d and σ_d are the mean and standard deviation, respectively, of the match population of s. This paper assumes a large n of at least several hundred allowing the previous property to hold approximately true. We therefore can produce a sampling of matches of s by randomizing each time point in s to create $\hat{s}$ such that:

$$s(i) - \hat{s}(i) \sim N(0, \sigma_r^2), \tag{3}$$

where σ_r is the standard deviation of the randomization of s. We define Y to be an independent random variable drawn from $dist(s,\hat{S})$ and X to be an independent random variable drawn from $s(i) - \hat{s}(i)$. Y is therefore distributed as:

$$Y = \sqrt{\sum_{i=0}^{n} X_i^2} = \sigma_r \chi_n \tag{4}$$

This yields a mean and standard deviation of:

$$\mu = \sigma_r \mu_{\chi_n} = \sigma_r \sqrt{2}\frac{\Gamma(\frac{n+1}{2})}{\Gamma(\frac{n}{2})} \tag{5}$$

$$\sigma = \sigma_r \sigma_{\chi_n} = \sigma_r \sqrt{(n - \mu_{\chi_n}^2)}, \tag{6}$$

with $\Gamma(x)$ being the Gamma function defined as:

$$\Gamma(x) = \int_0^\infty t^x e^{-t} \mathrm{d}t \tag{7}$$

A lookup table for μ and σ can be found in Table III as these computations can suffer from rounding error. For large n, distances between s and elements in $\hat{S}$ are given by the following distribution:

$$Y \sim N(\sigma_r \mu_{\chi_n}, \sigma_r \sigma_{\chi_n}) \tag{8}$$

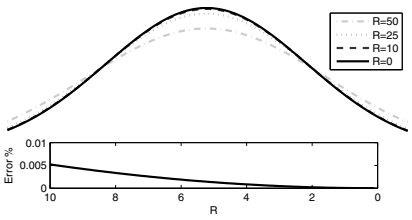

Figure 1. Shows the convergence of Equation 9 as $R \to 0$.

However, the likelihood that a randomized segment exists within a database is extremely low. This is especially true as both the granularity of points and length of time series (n) increase. Therefore, a Monte Carlo approximation is unlikely to yield reasonable results unless the number of samples is infinitely large. Therefore, all neighbors within a distance R to a randomized signal are used in the set of returned results. The probability of finding a segment s with distance in the range of $[x - R, x + R]$ to the search segment can be defined as:

$$f(z) = \frac{1}{2R\sqrt{2\pi\sigma^2}} \int_{x-R}^{x+R} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \, \mathrm{d}x \tag{9}$$

There is no closed form solution to Equation 9, however, as R approaches $0, f(z)$ approaches a Gaussian distribution. This paper assumes an R that is relatively small. Hence, the error introduced by LSH adds a negligible amount to the bounds of the proposed randomization approach. This is heuristically shown in Fig. 1.

A. Complexity

As stated earlier, the algorithmic complexity of LSH is sub-linear and dominated by $O(N^\rho)$ distance computations (pruning) where $\rho = \frac{\ln 1/p_1}{\ln 1/p_2}$ and N being the size of the database. Assuming a segment size of n, the randomization process is $O(mn)$ where m is the number of randomizations. LSH is run m times yielding a complexity dominated by $O(mN^\rho)$ (as $O(mN^\rho) > O(mn)$). This is still sub-linear as $m \ll N$.

B. Trivial Matches

This paper uses the formal definition of trivial matches proposed in [10]. Two segments $\hat{i} = [i, i + m']$ and $\hat{j} = [j, j + m']$ are trivial matches if $dist(\hat{i}, \hat{j}) \leq thr$ and there is no segment $\hat{i'} = [i', i' + m']$ where $i \leq i' \leq j$ and $dist(\hat{i}, \hat{i'}) > thr$ [10]. Trivial matches add redundancy to the result sets as they represent the same patterns with small translations. This paper probabilistically filters trivial matches by removing any segments that lie immediately next to or one time point away from a previously extracted segment. More formally, given a segment u_i returned by an LSH search, the segments u_{i-1}, u_{i-2}, u_{i+1}, u_{i+2} will be filtered as trivial matches (where i is the starting point of segment u). In addition, all trivial matches to previously declared trivial matches will also be removed. The observation is that trivial matches occur in small runs of neighboring segments. The probability of LSH missing one neighboring trivial match (p_2) is reasonably high in practice, but the probability of missing two neighboring segments(p_2^2) is quite low.

C. Filtering Optimization

LSH returns all segments with colliding hashes. The result set may contain several false positives. These false positives are filtered by taking the Euclidean distance between the query segment and all matches returned by LSH. Any segment with distance greater than R to the query segment is filtered. During experimentation, it was found that a large proportion of time was spent pruning false positives. Each dataset consists of tens of millions of indexed segments, and therefore, even a small percentage of false positives results is a large number of segments. This is especially true when aggregated over several random query segments.

To improve performance, results were pruned using the original query segment and not the randomized query segment. Note, the original algorithm searches and prunes results for each randomized segment independently. In this optimization, results were kept with probability based on their distances to the original query segment using the distribution $Y \sim N(\sigma_r \mu_{\chi_n}, \sigma_r \sigma_{\chi_n})$.

This pruning technique also improved the results for LSH (deemed LSH with sampling). Therefore, this paper compares the Monte Carlo approximation with both LSH and LSH with sampling. To note, LSH result sets could be improved by raising the value of R and running LSH with sampling to avoid the overhead added by the Monte Carlo scheme. However, the search runtime using LSH is heavily dominated by the number of distance comparisons during pruning. The Monte Carlo approximation raises the theoretical bound of pruning by a factor of m. On the other hand, raising R exponentially increases the theoretical bound. With the assumption of a uniformly distributed search space, raising R increases the search space by R^d (where d is the number of dimensions).

IV. Experimental Setup

This paper leveraged four datasets in its assessment of the proposed method: MIT-BIH Arrhythmia Database [11] (ECG), Gait Dynamics in Neuro-Degenerative Disease Database [12], [13] (GAIT), Synthetic Signal (SYN), and Synthetic Shapes (SHAPE). SYN was created using the function $f(x) = \sum_{i=3}^{7} \frac{1}{2^i} * \sin(\pi x 2^{3+i}) + rand(1)$. This is a derivative of the pseudo periodic synthetic time series data set presented in [14]. SHAPE was created to assess precision and recall and is composed of ten different shapes with varying levels of Gaussian noise displayed in Fig. 2.

Each dataset was indexed and inserted into the the database. A total of 50 segments were randomly chosen as

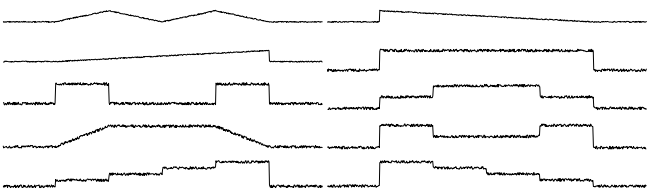

Figure 2. Displays the ten shapes (classes) in the synthetic shape dataset with Gaussian noise.

Table II
DATASET ATTRIBUTES

Dataset	Number of segments	Segment length	R
ECG	70M	512	3
GAIT	15M	512	3
SYN	10M	128	3
SHAPE	1M	600	1.5

query segments from each dataset. These segments were searched using LSH, LSH with sampling (proposed in Section III-C), and the full Monte Carlo method proposed by this paper. Both the standard deviation of the randomization (σ_r) and the number of randomizations (m) were varied for the Monte Carlo experiments. The attributes of each dataset are given in Table IV. The radius R is relative to the normalized distances (e.g., segments are normalized before finding the Euclidean distance).

The percentage improvements of result sizes of the Monte Carlo approximation over LSH and LSH with sampling are shown. The percentage increase is used as a comparison in lieu of raw counts due to the high variability in the size of result sets. The number of results is highly dependent on the random segment extracted from the time series signal. For example, a common heart beat from the ECG signal will generally match to several thousand segments. An anomaly, on the other hand, may only match to a couple of segments.

Precision and recall was assessed only for the SHAPE dataset with varying amount of Gaussian noise. The ECG, GAIT, and SYN datasets are not annotated to label specific neighborhoods. In addition, these datasets have tens of millions of rows and manually creating these labels is not feasible. However, the respective means and standard deviations of the Euclidean distance between the result sets and the query segment are given to show correctness. In addition, example result sets are displayed visually for each dataset for further evidence of correctness.

The runtime of the Monte Carlo approximation was also evaluated. This experiment consisted of 50 random runs using both LSH and the Monte Carlo approximation. Wall clock time (in seconds) is presented. All experiments were run on an Intel(R) Core(TM) i5 CPU 650 processor clocked at 3.2 GHz [15] running Ubuntu 11.10 [16].

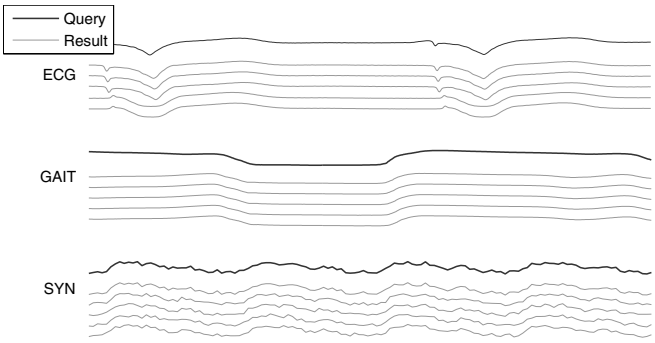

Figure 3. Displays three example query results for each dataset using the Monte Carlo approximation. Five segments were randomly displayed from each result set. The greatest variation between segments can be seen in the second column of ECG and the three columns of SYN.

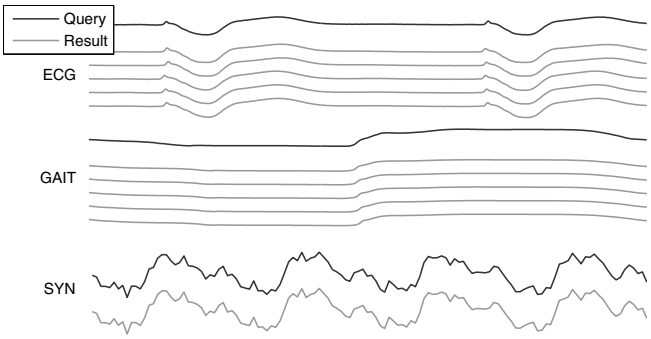

Figure 4. Displays example query results for the ECG, GAIT, and SYN datasets using LSH. Five segments were randomly displayed from each result set (when applicable). Almost no variation between segments is observed. The SYN dataset has a result size of one for all queries as shown by the figure.

V. RESULTS

Example Monte Carlo query results are shown for the ECG, GAIT, and SYN datasets in Fig. 3. Five segments are randomly extracted and displayed. The greatest variation between segments can be seen by the ECG and SYN datasets. This is unlike LSH where each result set contained almost no variability, as shown in Fig. 4. In fact, LSH consistently retrieves a dataset size of 1 (exact match) for the SYN dataset. The poor performance of LSH for SYN is due to the randomization added during the construction of SYN. This phenomenon is important as the randomization of the dataset SYN is similar to the effects of noise. The poor results for SYN demonstrates the degradation of LSH's performance with signal noise.

The percentage increase in the number of results over standard LSH when varying the randomization standard deviation (σ_r) and the number of randomizations (m) is shown in Fig. 5. The results for SYN with varying σ_r are shown separately from the ECG and GAIT datasets as the increase in result size is several orders of magnitude

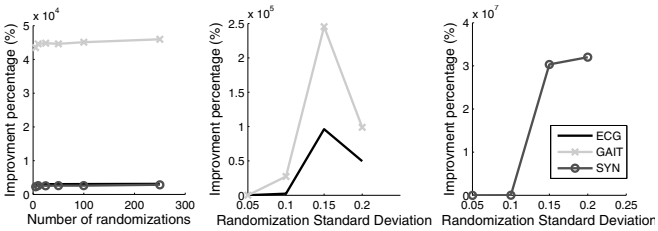

Figure 5. Displays the percentage increase in the results sets over standard LSH when varying both m and σ_r. Both a fixed $\sigma_r = .1$ with $m = 5, 10, 25, 100, 250$ and a fixed $m = 25$ with $\sigma_r = .05, .1, .15, .2$ are displayed. The results for SYN with varying σ_r is shown separately from the ECG and GAIT datasets as the increase in result size is several orders of magnitude larger.

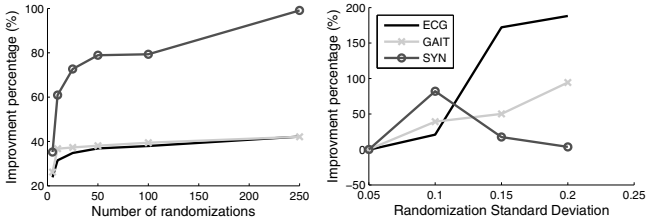

Figure 6. Displays the percentage increase in the results sets over LSH with sampling when varying both m and σ_r. Both a fixed $\sigma_r = .1$ with $m = 5, 10, 25, 100, 250$ and a fixed $m = 25$ with $\sigma_r = .05, .1, .15, .2$ are displayed.

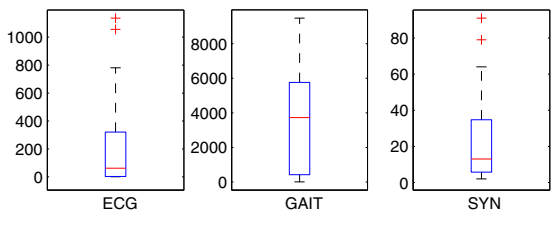

Figure 7. Displays the number of results for 50 randomly selected queries with $m = 25$ and $\sigma_r = .1$.

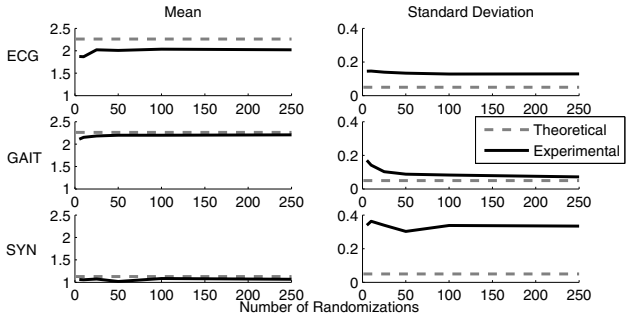

Figure 8. Displays the man and standard deviation of the Euclidean distance from query segments to the respective result sets. Queries were run with $\sigma_r = .1$ and $m = 5, 10, 25, 100, 250$. The respective theoretical means and standard deviations are also displayed.

larger. The increase over standard LSH ranged from a couple hundred percent to several thousands. This is expected as standard LSH only returns segments that are very similar to the query segment. The larger increase by the GAIT dataset when varying m is caused by the homogeneity of the dataset (i.e., each step in the dataset is largely similar). The randomizations do a good job of expanding the search space. This is extremely important as this same phenomenon would also be seen in other datasets as the number of indexed segments increases.

Increasing σ_r exhibits larger increases for the SYN dataset over standard LSH. A larger σ_r models a more significant amount of noise. The GAIT and ECG datasets are reasonably clean, and therefore, do not benefit from a larger σ_r. In fact, both the results sets for GAIT and ECG decreased with $\sigma_r > .15$. Larger σ_r cause the search space to extend beyond the neighborhood of the query segment resulting in fewer matches. The SYN dataset, on the other hand, has a large amount of noise. Hence, a larger σ_r helps improve the size of result sets (as the neighborhood is larger).

The percentage increase over LSH with sampling of result set sizes when varying σ_r and m is shown in Fig. 6. Most of this improvement is seen with less than 50 randomizations.

As stated earlier, the size of a result set is highly dependent upon the respective query segment. A common pattern will yield a large number of results, while an anomaly will yield small sets. Therefore, both Fig. 5 and Fig. 6 displayed percentage increases. Fig. 7 displays raw result set sizes with $m = 25$ and $\sigma_r = .1$ for a reference.

The goodness of the result sets for the Monte Carlo approximation was assessed by finding the mean and standard deviation of the Euclidean distance of the query segments to their respective result segments. Fig. 8 and Fig. 9 display the effects of varying both m and σ_r respectively. When increasing m, the mean and standard deviation converges to the respective theoretical values with the exception of standard deviation for SYN. The SYN dataset's higher standard deviation is due to the addition of uniformly distributed noise that results in neighborhoods that are not normally distributed. A slight divergence in standard deviation is observed when increasing σ_r. This is expected as increasing σ_r will increase the search outside of a segments immediate neighborhood.

Precision and recall results for the SHAPE dataset are shown in Table V. All tests resulted in a 100% precision. This is expected as the neighborhoods are known with a radius $R < 1.5$ with high discrimination from other classes. Hence, setting an accurate neighborhood will assure precise results. Recall for LSH with sampling is approximetely 91% while recall for Monte Carlo search is near 100% with as little as ten randomizations ($m = 10$).

VI. Conclusions

This paper presents a Monte Carlo approximation technique for subsequence matching. The number of results for the Monte Carlo approximation are significantly increased

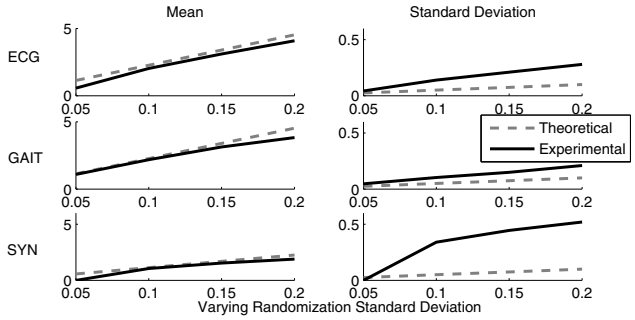

Figure 9. Displays the mean and standard deviation of the Euclidean distance from query segments to the respective result sets. Queries were run with $\sigma_r = .05, .1, .15, .2$ and $m = 25$. The respective theoretical means and standard deviations are also displayed.

Table III
PRECISION RECALL FOR LSH WITH SAMPLING AND MONTE CARLO SEARCH ($m = 10$)

Method	Precision	Recall
LSH	1.0	.909089
LSH with sampling	1.0	.912844
Monte Carlo	1.0	.999978

over standard Locality Sensitive Hashing (LSH). This technique adds minimal computational complexity and ensures the preciseness of results. The proposed technique takes in a subsequence as input. The subsequence is randomized m times using a Normal distribution with standard deviation equal to σ_r. The resulting m randomized sub-sequences are provided as input to LSH resulting in m distinct R-NN searches. It was theoretically and experimentally shown that the Euclidean distances of a result set to the respective query segment is bounded by a Normal distribution of $Y \sim N(\sigma_r \mu_{\chi_n}, \sigma_r \sigma_{\chi_n})$.

ACKNOWLEDGMENT

The authors would like to thank Alexandr Andoni et. al. for providing their implementation of LSH presented in [9]. This publication was partially supported by Grant Number T15 LM07356 from the NIH/National Library of Medicine Medical Informatics Training Program.

REFERENCES

[1] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" *Database Theory-ICDT'99*, pp. 217–235, 1999.

[2] J. Woodbridge, M. Lan, M. Sarrafzadeh, and A. Bui, "Salient segmentation of medical time series signals," in *Healthcare Informatics, Imaging and Systems Biology (HISB), 2011 First IEEE International Conference on*. IEEE, 2011, pp. 1–8.

[3] K. Bennett, U. Fayyad, and D. Geiger, "Density-based indexing for approximate nearest-neighbor queries," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 233–243.

[4] M. Houle, H. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Can shared-neighbor distances defeat the curse of dimensionality?" in *Scientific and Statistical Database Management*. Springer, 2010, pp. 482–500.

[5] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the International Conference on Very Large Data Bases*, 1999, pp. 518–529.

[6] R. Weber, H. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the International Conference on Very Large Data Bases*. IEEE, 1998, pp. 194–205.

[7] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '94. New York, NY, USA: ACM, 1994, pp. 419–429. [Online]. Available: http://doi.acm.org/10.1145/191839.191925

[8] Y. Cai and R. Ng, "Indexing spatio-temporal trajectories with chebyshev polynomials," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM, 2004, pp. 599–610.

[9] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004, pp. 253–262.

[10] J. Lonardi and P. Patel, "Finding motifs in time series," in *Proc. of the 2nd Workshop on Temporal Data Mining*, 2002, pp. 53–68.

[11] R. Mark and G. Moody, "Mit-bih arrhythmia database directory," *Cambridge: Massachusetts Institute of Technology*, 1988.

[12] J. Hausdorff, A. Lertratanakul, M. Cudkowicz, A. Peterson, D. Kaliton, and A. Goldberger, "Dynamic markers of altered gait rhythm in amyotrophic lateral sclerosis," *Journal of applied physiology*, vol. 88, no. 6, pp. 2045–2053, 2000.

[13] J. Hausdorff, S. Mitchell, R. Firtion, C. Peng, M. Cudkowicz, J. Wei, and A. Goldberger, "Altered fractal dynamics of gait: reduced stride-interval correlations with aging and huntington's disease," *Journal of Applied Physiology*, vol. 82, no. 1, p. 262, 1997.

[14] S. Park, D. Lee, and W. Chu, "Fast retrieval of similar subsequences in long sequence databases," in *Knowledge and Data Engineering Exchange, 1999.(KDEX'99) Proceedings. 1999 Workshop on*. IEEE, 1999, pp. 60–67.

[15] Intel. (2011) 2nd generation intel core processor family desktop datasheet. [Online]. Available: http://www.intel.com/content/www/us/en/processors/core/2nd-gen-core-desktop-vol-2-datasheet.html

[16] R. Petersen, *Ubuntu 11. 10 Desktop: Applications and Administration*. Surfing Turtle Press, 2011.