liteDTW: A Lightweight Dynamic Time Warping for Tiny Wireless Sensing Devices

Mohamed Abdelaal, Peter Nazier Mosaad, Oliver Theel
Carl von Ossietzky University of Oldenburg, Oldenburg, Germany
{mohamed.abdelaal | peter.nazier.mosaad | theel}@informatik.uni-oldenburg.de

***Abstract*—Wireless sensor networks have been recognized as promising tools to collect relevant, in-situ data for a wide range of application domains. However, such networks suffer from hard constraints including the allocated resources. Hence, current research endeavors strive to minimize the amount of data that has to be transmitted. This is typically achieved via data fusing or sending some nodes to sleep mode whenever their readings exhibit a high degree of spatio/temporal correlation. Accordingly, the degree of correlation can be considered as a metrics for data filtering. The *Dynamic time warping* (DTW) algorithm is a "natural" candidate for data fusion and correlation estimation at intermediate sensor nodes via matching the various measured readings. However, the DTW algorithm suffers from the excessive computational overhead which is ill-suited for the "resources-taxed" sensor nodes. This work aims at reducing this burden via refining the implementation procedure of the DTW algorithm. The *liteDTW* is a novel version of the DTW algorithm with linear operations and fuzzy abstraction. The core idea is to reduce the DTW matrix dimensions via shrinking the input patterns. Several simulations and real experiments have been conducted to validate that the *liteDTW* algorithm excels over the naïve one in terms of accuracy, time and space overhead. Moreover, the Cooja simulator of the Contiki OS has been utilized to assess the energy profit of adopting the *liteDTW* algorithm for data fusion.**

Keywords*—Wireless Sensor Networks; Energy Efficiency; Data Fusion; Dynamic Time Warping; Fuzzy Abstraction; Complexity

I. Introduction

Wireless sensor networks (WSNs) paved their way as a fertile research space with a vivid business window. Their markets at $552.4 million in 2012 become extremely dilated, very fast reaching $14.6 billion by 2019 [1]. The WSN development has been driven by several applications such as military applications, industrial process monitoring and control machine health monitoring, and so on. Data redundancy is an undesired issue in WSN applications, whenever energy efficiency is of high interest. Data fusion and duty cycling are typically two methods for eliminating such redundancies. Data fusion is generally defined as the use of techniques that merge information from multiple nodes and gathering this information to achieve inferences [2]. Moreover, data fusion techniques are recently utilized to overtake sensor failures, spatial and temporal coverage problems, and technological limitations. Duty cycling, on the other hand, is a well-known approach to reduce data redundancy via sending unnecessary nodes to sleep mode. The crux here is to exploit the spatio/temporal correlation that exists among neighboring sensor nodes. Specifically, aggregating nodes have to discover the correlation degree between the various received readings. Afterward, they decides about the redundant nodes which could be switched to sleep mode. Driven by the demand in WSNs for using data fusion and duty cycling techniques, we have recently seen a rapidly growing interest in developing the way to process the collected data and addressing the pattern matching problem. The pattern matching problem has been previously addressed in the context of data mining [3]. Here, we will focus on solving the pattern matching problem by using the dynamic time warping (DTW) algorithm.

Dynamic time warping is a well-known technique to find an optimal alignment between two sequences under certain restrictions [4]. It has been widely used for optimal alignment of two time series through iteratively warping the time axis until an optimal match between the two sequences is found. However, such an algorithm is bulky and ill-suited for the tiny sensor nodes. In this paper, the DTW implementation is refined to minimize the time/space complexity. We call our novel algorithm *lightweight dynamic time warping* (*liteDTW*). The main idea of *liteDTW* is to 1) reduce the required memory footprint, and 2) simultaneously shrink the length of the input patterns. For the sake of linear implementations, the DTW matrix is evaluated over several iterations. At each iteration, two columns of the matrix are solely considered to determine the optimal path points. Then, shift operations are executed to iterate over the rest of the DTW matrix. The second approach is to slash the window size via using a Fuzzy transform-based compression (FTC) technique [5]. Accordingly, the data dimension is decreased prior to the DTW execution. Finally, some experimental results are conducted to clarify the efficiency of the *liteDTW* technique compared to the classical DTW algorithm.

The remainder of this paper is organized as follows. Section II provides a brief overview of existing DTW speed-up methods. Section III discusses the basic idea behind DTW algorithm. Moreover, the DTW algorithm is evaluated in terms of accuracy and cost. Section IV depicts two approaches for mitigating the DTW complexity including algorithm linearization and FTC-based abstraction. Section V presents the *liteDTW* performance evaluation through simulations as well as real experiments. Finally, conclusion and outlook are addressed in Section VI.

II. Related Work

In literature, several articles have discussed the DTW algorithm as in [6], [7], [4]. In the last decades, a lot of

This research is funded by the German Research Foundation through the Research Training Group DFG-GRK 1765: "System Correctness under Adverse Conditions" (SCARE, scare.uni-oldenburg.de).

attention has gone to modify the classical DTW algorithm as it has a quadratic time and space complexity that limits its use to only small time series data sets. The well-established, yet still being subject of active research, techniques for making the DTW faster fall into three categories, *constraints* [8], [9], *data abstraction* [10], [11], and *indexing* [12], [13]. Constraints typically limit the number of cells that are evaluated in the cost matrix. In the second category, data abstraction performs the DTW algorithm on a reduced representation of the data. Finally, indexing uses lower bounding functions to shrink the number of times the DTW algorithm must be executed during time series classification of clustering. The proposed techniques in [14] and [15] are promising techniques with some similarities to our work.

Salvador et al. [14] proposed an approximation of the DTW algorithm based on the multilevel approach that is used for graph bisection. The algorithm uses techniques that belong to the two categories, namely constraints and data abstraction. For instance, the algorithm uses projection operation to find the minimum-distance warp path at a lower resolution, and it uses that warp path as an initial guess for a higher resolution's minimum-distance warp path. However, our method linearizes the DTW algorithm by only retaining two columns each iteration. Therefore, we avoid the disadvantages of projecting the warp path from low resolution to higher resolution. This projection may ignore local variations in the warp path that can be very significant even after refinement. Moreover, we use Fuzzy transform-based compression (FTC) to lessen the data dimension due to its high speed and adequate precision. This abstraction is more efficient for tiny devices such as in WSNs than the coarsening operation presented in [14].

Sakurai et al. [15] proposed a fast search method for the DTW algorithm. The core idea is to use a lower bounding distance measure with segmentation (LBS) technique. In lieu of computing the exact time warping distance for all sequences in the dataset, LBS prunes a significant number of sequences. Then, it excludes warping paths that will not lead to useful search result by using dynamic programming. Finally, the authors have used a search algorithm to enhance the accuracy of the distance approximations. This technique is significant, however our technique is much simpler and easy to implement especially in case of WSN applications. In addition, our approach has been examined on real datasets via extensive simulations over the Telosb sky sensor nodes. In the next section, we introduce the basics of patterns matching using the DTW algorithm.

III. Dynamic Time Warping

A. Preliminaries

As aforementioned, the pattern matching problem has been previously addressed in the context of data mining [3]. *Euclidean distance* is a well-known linear metric for quantifying the distance between two vectors. Assume two sequences, $A \in \mathbb{R}^n$ and $B \in \mathbb{R}^m$ to be correlated where

$$A = \langle a_1, a_2, a_3, \ldots, a_i, \ldots, a_n \rangle \tag{1}$$

$$B = \langle b_1, b_2, b_3, \ldots, b_j, \ldots, b_m \rangle \tag{2}$$

the Euclidean distance $Euc(A, B)$ is as denoted as follows.

$$Euc(A, B) = |A - B| \qquad \text{iff} \quad m = n \tag{3}$$

Despite its simplicity, the Euclidean metrics is ill-suited for real-time tasks with sequences of unequal lengths. Moreover, it is highly sensitive to outliers. The *dynamic time warping* (DTW) algorithm, on the other hand, is a classic time series alignment algorithm. It has been widely used for optimal alignment of two time series through warping the time axis iteratively until an optimal match (according to a suitable metrics) between the two sequences is found. In general, the DTW's non-linear behavior produces a more intuitive similarity measure compared with the Euclidean distance. As shown in Fig. 1, the DTW measure replaces the one-to-one point comparison, used in the Euclidean distance, with a many-to-one (and vice versa) comparison. The green lines represent mapping between points of time series T and S. The main feature of this distance measure is that it allows to recognize similar shapes, even if they present signal transformations, such as shifting and/or scaling.

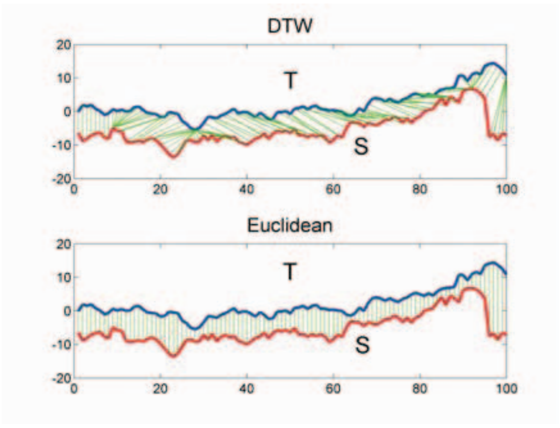

Fig. 1: Comparing the behavior of the DTW and the Euclidean distances [16]

Figure 2 visualizes the matching between a reference and a test pattern arranged on the sides of a $m \times n$ matrix where the elements are the DTW distances $d_{n,m}$ as expressed in Eq. 4. Several paths could be drawn from element $(1, 1)$ to element (n, m) of the matrix. However, the optimum alignment $P_{opt} = \langle p_1, p_2, \ldots, p_k \rangle$ minimizes the total inter-distances as denoted by Eq. 5.

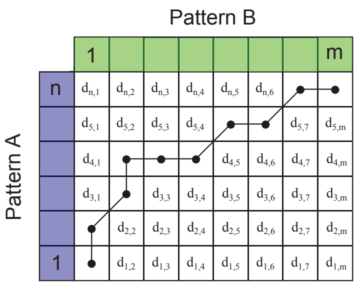
Pattern B
Pattern A
1
m
n
1
d_{n,1} d_{n,2} d_{n,3} d_{n,4} d_{n,5} d_{n,6}
d_{5,1} d_{5,2} d_{5,3} d_{5,4} d_{5,7} d_{5,m}
d_{4,1} d_{4,5} d_{4,6} d_{4,7} d_{4,m}
d_{3,1} d_{3,3} d_{3,4} d_{3,5} d_{3,6} d_{3,7} d_{3,m}
d_{2,2} d_{2,3} d_{2,4} d_{2,5} d_{2,6} d_{2,7} d_{2,m}
d_{1,2} d_{1,3} d_{1,4} d_{1,5} d_{1,6} d_{1,7} d_{1,m}

Fig. 2: Choosing the optimum warping path of length $k = 10$

$$d_{n,m} = \begin{cases} |a_1 - b_1| & \text{if } n = m = 1 \\ |a_n - b_m| + W_{n,m} & \text{otherwise} \end{cases} \qquad (4)$$
$$W_{n,m} = min(d_{n-1,m}, d_{n,m-1}, d_{n-1,m-1})$$

$$P_{opt} = \underset{P}{min} \left\{ \sum_{s=1}^{k} d_{n,m} \right\} \qquad (5)$$

The search space is governed by a set of design constraints summarized in Table I. First, the path P should continuously advance one-step at a time to avoid discarding important features. Moreover, the path should be monotonically non-decreasing to hamper features recurrence. Finally, the start and end points should extend from point $(1, 1)$ to point (n, m) to align the two sequences. In some applications, a global rule defines a warping window $R \subseteq [1 : n] \times [1 : m]$ to speed up the algorithm. Nevertheless, confining the search space to the window R is debatable, since the path P_{opt} may traverse cells outside the specified constraint region. Thereof, we deliberately ignored this constraint for matching optimization.

TABLE I: A summary of alignment path constraints

Continuity	$d_{i,j} - d_{i-1,j} \leq 1$ & $d_{i,j} - d_{i,j-1} \leq 1$
Monotonicity	$d_{i-1,j} \leq d_{i,j}$ & $d_{i,j-1} \leq d_{i,j}$
Boundary Conditions	$p_1 = d_{1,1}$ & $p_k = d_{n,m}$

Algorithm 1 shows a description of our recursive implementation of the DTW algorithm. In lieu of utilizing two matrices to determine the DTW distance, we modified the procedure in a recursive way to only process one matrix. At the outset, the DTW distances are horizontally and vertically determined. Then, Eq. 4 evaluates the rest of the matrix. Through lines 9-15, the path P_{opt} – which is initialized to the empty path – is selected through point-to-point optimization. Since the DTW has to compare several sensor patterns, each DTW distance is normalized to the length of P_{opt}, denoted by k, enabling fair comparisons. To determine the best match, the normalized distance χ_{min} is chosen as given in line 16. Equation 6, on the other hand, defines a threshold H via normalizing the χ_{min} distance. This margin defines a binary similarity decision between two patterns. Such a decision results in activating or deactivating sensor nodes with highly correlated data. Moreover, Eq. 6 normalizes the distance $DTW(A, B)$. Hence, H is not updated in accordance with ϕ. Subsequently, we show experimentally (Sec. III-B) that a fixed value of H = 0.1 is robust across a wide range of patterns. Next, we evaluate the performance of the standard DTW in terms of its accuracy and cost via using real measured vibration patterns.

$$[\chi_{min}] \cdot [\chi_{max}]^{-1} < H \qquad (6)$$

B. DTW Evaluation

In this section, DTW precision and time/space complexity are examined. For this purpose, an Arduino UNO board has been utilized to sample seismic patterns from a LDT piezoelectric vibration sensor. Different measuring scenarios of speed 0.5 m/sec have been considered and summarized in Table II. For comparison purposes, the vibration patterns are classified into target T and non-target NT patterns.

Algorithm 1 Recursive implementation of the DTW algorithm

```
Require: Reference pattern $A \in \mathbb{R}^n$, and test pattern $B \in \mathbb{R}^m$
 1: for $i$ such that $1 \leq i < n$ do
 2:     for $j$ such that $1 \leq j < m$ do
 3:         $d_{1,1} \leftarrow |a_1 - b_1|$;
 4:         if $!(i)\& \ j == 1$ then                      ▷ Horizontal border
 5:             $d_{1,j} \leftarrow \sum_{x=1}^{j}(|a_1 - b_x|)$;
 6:         else if $i$ & $!(j) == 1$ then                  ▷ Vertical border
 7:             $d_{i,1} \leftarrow \sum_{y=1}^{i}(|a_y - b_1|)$;
 8:         else  $d_{i,j} \leftarrow |a_i - b_j| + W_{i,j}$;      ▷ Matrix's heart
 9: while $i < 1$ & $j < 1$ do                              ▷ Optimal path
10:     $P_{opt} \leftarrow P_{opt} + d_{i,j}$;
11:     if $W_{i,j} == d_{i-1,j}$ then
12:         $i \leftarrow (i - 1)$;
13:     else if $W_{i,j} = d_{i,j-1}$ then
14:         $j \leftarrow (j - 1)$;
15:     else  $i \leftarrow (i - 1)$ & $j \leftarrow (j - 1)$ ;
16: $\chi(A, B) \leftarrow \frac{P_{opt}}{k}$;                     ▷ Normalization
```

TABLE II: Indexing reference and test patterns

Index	Target Patterns	Index	Non-target Patterns
T_1	Straight walking (indoor)	NT_1	Background
T_2	Straight walking (outdoor I)	NT_2	Finger drumming
T_3	Straight walking (outdoor II)	NT_3	Straight walking by a mobile machine (indoor)
T_4	Circle walking (indoor)	NT_4	Straight walking by a mobile machine (outdoor)
T_5	Circle walking (outdoor)		

Figure 3 depicts samples of DTW accuracy results obtained from contrasting some targeted and non-targeted vibration patterns. In each sub-figure, a vibration pattern is compared to the other patterns. Knowing that $DTW(A, A) = 0$, pattern A_{indoor} is matched with $A_{outdoor}$ to clarify the process of selecting the best match. Obviously, the DTW algorithm has successfully matched indoor and outdoor pairs via adopting the minimum DTW distance. A fixed margin of $H = 0.1$ (colored in pink) is utilized to make a decision of similarity between two patterns whenever the distance exceeds it.

C. Cost Analysis

Generally, manifold cells of a $m \times n$ matrix are filled exactly once throughout the DTW execution, and each cell is filled in constant time. This yields both a space complexity of $\mathrm{O}\big(n \times m\big)$. Initially, the window size of the contrasted patterns is a main metric which affects the algorithm speed and the memory footprint. Figure 4 depicts the impact of increasing the window size on the distance between the pattern $T1$ and some other patterns. As it can be seen, the pattern $T2$ has minimum distance with $T1$ for the entire range of window sizes. So, we conclude that the normalized distance – in these settings – is distinguishable even with small windows.

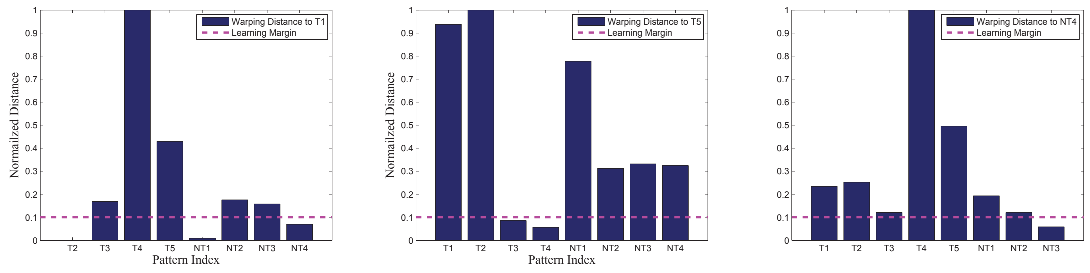
Fig. 3: Examples of the DTW algorithm utilization for vibration pattern matching

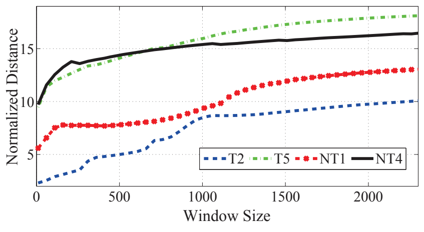
Fig. 4: Window size versus warping distance from T1

However, the above finding is collapsed when other patterns are examined versus various window sizes to safely adopt a window size that is adequate for matching the entire patterns. Figure 5 depicts the minimum window size for each pattern to be distinguishable in a duty cycling approach. For instance, the pattern $NT2$ can be matched easily if the window size is above 200. Obviously, sampling ϕ up to the maximum value (960) ensures safe detections for all vibration patterns. However, this window size burdens the nodes in terms of space/time complexity. To sum up, the DTW-based duty cycling will be solely eligible for hardware implementation, whenever we managed to slash the optimal window size.

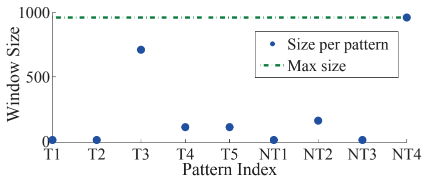
Fig. 5: Optimal window size for the available patterns

IV. liteDTW: DTW Refinement

In this section, we explain our proposed procedure for minimizing the time/space complexity from $O(n \times m)$ to an extent viable for hardware implementation. The idea is to integrate two complementary approaches: one for reducing the code complexity and memory utilization, and the other one for slashing the window size. Both approaches, as discussed below, upgrade the standard DTW algorithm into a new version referred to as the *liteDTW* algorithm.

1) Linear DTW Algorithm: The first approach is to linearize the time/space complexity implementation of the DTW algorithm. This is feasible through preserving only the current and previous columns in memory as the DTW matrix is filled from left to right. Figure 6 shows a three-iterations matching process between two sequences. In each iteration, only two columns are retained and points of the optimal warp P_{opt} (colored in red) are determined. Then, the first column is discarded, while the second column is used to estimate the third column. This process is repeated until covering the entire matrix. Algorithm 2 formalizes the linearization mechanism. Lines 2-5 clarify the first two columns' processing. Afterward, the $(n\times2)$ sub-matrix (colored green in Fig. 6) is shifted once to discard the first column and the variable ρ is set to 1 to compute only one column during the next iteration. Specifically, the linear DTW method simplifies the execution overhead from $O(n \times m)$ to merely $O(n \times 2)$ which highly reduces the required memory footprint.

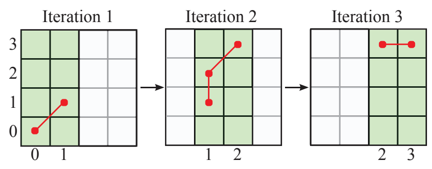

Fig. 6: Two-columns version of the DTW algorithm

Algorithm 2 Two-columns version of the DTW algorithm

```
Require: Reference pattern $A \in \mathbb{R}^n$, and test pattern $B \in \mathbb{R}^m$,
    $\rho = 0$
1: for $s$ such that $0 \leq s < m-1$ do          ▷ (m-1) iterations
2:     for $i$ such that $0 \leq i < n$ do
3:         for $j$ such that $\rho \leq j < 2$ do
4:             Determine $d_{i,j}$ as in Eq. 4
5:     Select $d_{i,j} \in P_{opt}$;
6:     $d[n \times 2] \leftarrow$ left_shift($d[n \times 2]$);
7:     $\rho \leftarrow 1$;                        ▷ Evaluating only one column
8: $\chi(A,B) \leftarrow \sum(P_{opt})/k$;
```

2) Fuzzy Abstraction: The crux here is to lessen the data dimension prior to DTW execution. Various techniques have been introduced in the literature for data compression. However, we prefer our Fuzzy transform-based compression due to its simplicity while offering adequate precision. The approximation error – introduced by the compression process – is relative and has no influence on the correlation decision since it is applied to the entire set of contrasted patterns.

The *Fuzzy transform* is defined as a fuzzy set mapper of a continuous/discrete function into an n-dimensional vector [17]. Assume that a time series is confined into an interval $\phi = [a, b]$ as a universe. This domain is fuzzy-partitioned by Fuzzy sets given by their basic function. Figure 7 depicts an example of a uniform triangular basic function with equidistant points given by Eq. 7. The red line implies that summation of any two vertical points should be equal to 1. Generally, the shape of basic functions forges the approximating function. Hence, the F-transform is well-suited for dealing with linear and non-linear sensor readings.

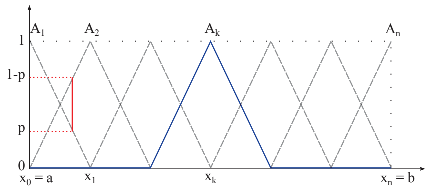

Fig. 7: Uniform triangular basic function

$$x_k = a + \frac{(b-a)(k-1)}{(n-1)} \tag{7}$$

Strictly speaking, the *direct F-transform* converts the original signal into an n-dimensional vector, where n corresponds to the number of triangular functions applied. *Inverse F-transform*, on the other hand, approximates the original signal utilizing the Fuzzy vector. The F-transform is explicitly defined for discrete as well as continuous functions.

Definition 1. *Assume a fuzzy partition of ϕ be given by basic functions $A_1, ..., A_n \subset \phi$ and $n > 2$. If a F-transformer has been triggered with a discrete function $f : \phi \rightarrow R$ known at nodes $x_1, ..., x_l$ such that for each $k = 1, ..., n$, there exists $j = 1, ..., l : A_k(x_j) > 0$. Then, the n-tuple of real numbers $[F_1, ..., F_n]$ is given by*

$$F_k = \frac{\sum_{j=1}^{l} f(x_j) A_k(x_j)}{\sum_{j=1}^{l} A_k(x_j)}. \tag{8}$$

The Fuzzy control theory is crucial for understanding the F-transform essence. Specifically, the direct F-transform resembles a defuzzification process (Center of gravity) through which linguistic variables ("low", "medium", "high", etc.) are mapped onto real numbers. This implies that each vector element F_k constitutes the weighted average of the data points $f(x_j) \in [x_{k-1}, x_{k+1}]$.

Definition 2. *Suppose a fuzzy vector $F_n[f] = [F_1, ..., F_n]$ w.r.t. $A_1, .., A_n$ has been applied to an inverse F-transformer. The recovered signal is given by*

$$f_{F,n}(x) = \sum_{k=1}^{n} F_k A_k(x). \tag{9}$$

The basic function's characteristics such as their shape and length, devote a fine-grained control over the recovery process. Therefore, they have to be carefully designed to avoid imperfect transformation. The interested readers can find more properties and proofs in [17].

V. Performance Evaluation

In this section, the *liteDTW* algorithm is evaluated and compared to the recursive procedure of the DTW algorithm. Two sets of evaluations have been conducted. The simulations have been devoted to evaluate accuracy, time, and space complexity of both algorithms in a sample setting. In order to validate the profit of adopting the *liteDTW* algorithm for data fusion, a network of TelosB sensor nodes has been simulated in a *Cooja* environment.

Both of DTW and *liteDTW* algorithms have been implemented using the C language. The simulator runs on a machine with 2.5 GHz processor and 8 GB RAM with Windows 7 OS. These kind of simulations are utilized to prove the excel of *liteDTW* over the naïve algorithm regarding time/space complexity. Moreover, the *liteDTW*'s accuracy is examined relative to the recursive algorithm. Figures 8 and 9 depict samples of comparison between the standard DTW algorithm and the *liteDTW* for comparing the patterns $NT4$ and $T1$ of Section III-B with other patterns utilizing a thousand data points. Obviously, *liteDTW* has an identical precision as the naïve DTW although *liteDTW* solely matches fifty fuzzy-compressed samples. For instance, both algorithms generate a minimum correlation between the patterns $T1$ and $T2$ as shown in Fig. 9. Nevertheless, *liteDTW* has a memory footprint of 800 bytes whereas the naïve DTW demands 7.6 MByte using the same data points. Thus, the *liteDTW* algorithm is an efficient tool for detecting correlations.

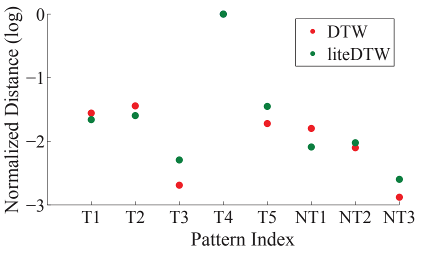

Fig. 8: Precision of liteDTW versus DTW for $NT4$ matching

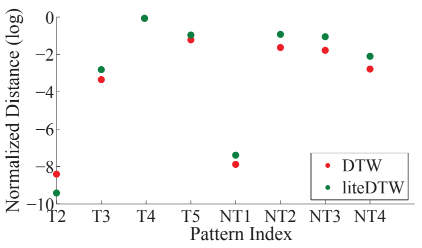

Fig. 9: Precision of liteDTW versus DTW for $T1$ matching

The execution time of an algorithm is another significant metrics for recognizing the algorithm's complexity. This set of experiments has not been run on typical sensor nodes – due to complexity of the naïve DTW. However, it gives us a "good" indication about the difference in complexity between these two implementations. Table III lists the obtained results of running the two algorithms. For the *liteDTW* algorithm, it has been tested for different compression ratios including $10\%, 25\%, 50\%,$ and 75%. Each of these delay values represents an average of several executions. Clearly, both algorithms have an exponential growth with increasing the window size. However, the *liteDTW* has much smaller delay than that of the naïve DTW even with low compression ratio.

TABLE III: CPU time consumption (in sec) of the DTW and the *liteDTW*

Window Size	DTW	liteDTW			
		10%	25%	50%	75%
200	0.43	0.01	0.02	0.03	0.06
400	0.82	0.02	0.04	0.11	0.24
600	1.88	0.03	0.06	0.26	0.55
800	3.25	0.03	0.12	0.42	0.98
1000	5.54	0.04	0.18	0.69	1.52
2000	21.77	0.13	0.69	2.63	6.11

In the sequel, the *liteDTW* is ported to an aggregating node which executes data fusion. The setup involved a network under the ConikiMAC radio duty cycling protocol with the *collect* communication primitive in the Rime stack of the Contiki OS. The run-time power consumption was estimated utilizing the *powertrace* module as denoted in Eq. 10. The power in milliwatt is a function of the activity (ψ) start and end times, the operating current (I) which can be obtained from the datasheet, the supply voltage (V_{supply}), the number of ticks per second (α), and the experiment runtime (τ) in seconds. Table IV lists the radio and CPU power consumption of the aggregating node in one round. The power consumption is determined for several network sizes (excluding the aggregating node). For instance, the aggregating node consumes 260.56 mW by the radio module for transmitting and receiving packets from different 4 child nodes. Alternatively, it can only consumes 135.98 mW if the *liteDTW* algorithm is adopted. To sum up, radio power consumption is drastically reduced with data aggregation at the expense of slight increase in the CPU power consumption.

TABLE IV: Power consumption (in mW) of an aggregating sensor node

Network Size	Without data fusion		With data fusion	
	Radio	CPU	Radio	CPU
2	130.29	12.16	130.29	24.33
3	195.44	19.12	119.59	30.37
4	260.59	31.25	135.98	41.02
5	325.74	35.68	104.98	58.43

$$P_{\psi} = \frac{(\psi_{end} - \psi_{start}) \times I \times V_{supply}}{\alpha \times \tau} \tag{10}$$

VI. Conclusion

In this work, a novel approach, referred to as *liteDTW*, has been proposed to speed up the DTW pattern matching algorithm. It utilizes the FTC compressor for reducing the input pattern lengths. Hence, the number of cells in the DTW matrix is drastically reduced. Moreover, the *liteDTW* algorithm reduces the required memory footprint via dividing the implementation process into several iterations. Thus, the complexity is reduced from $\mathrm{O}(n^2)$ to only $\mathrm{O}(2n)$. Accordingly, the *liteDTW* algorithm has been shown as a comprehensive solution for solving the redundancy problem via data fusion and duty cycling. A set of experiments proved that the *liteDTW* is an efficient method in terms of accuracy, time/space complexity, and energy consumption. For the future work, we plan to conduct an experimental comparative analysis between the *liteDTW* and other speeding up mechanisms.

References

[1] "Wireless Sensor Networks: Market Shares, Strategies, and Forecasts, Worldwide, 2013 to 2019," accessed on Aug. 2014. [Online]. Available: http://goo.gl/q5JKbS

[2] A. Abdelgawad and M. Bayoumi, *Resource-Aware data fusion algorithms for WSNs*. Springer Science & Business Media, 2012.

[3] M. Muller, *Information Retrieval for Music and Motion*. Springer, 2007, ch. Dynamic Time Warping. [Online]. Available: http://goo.gl/FUdsyi

[4] M. Müller, *Information retrieval for music and motion*. Springer, 2007.

[5] M. Abdelaal and O. Theel, "An Efficient and Adaptive Data Compression Technique for Energy Conservation in Wireless Sensor Networks," in *2013 IEEE Conference on Wireless Sensor (ICWISE)*, Dec 2013, pp. 124–129.

[6] M. Parizeau and R. Plamondon, "A comparative analysis of regional correlation, dynamic time warping, and skeletal tree matching for signature verification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 7, pp. 710–717, 1990.

[7] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping." in *SDM*, vol. 1. SIAM, 2001, pp. 5–7.

[8] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, 1978.

[9] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 23, no. 1, pp. 67–72, 1975.

[10] S. Chu, E. J. Keogh, and et al., "Iterative deepening dynamic time warping for time series." in *SDM*. SIAM, 2002, pp. 195–212.

[11] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 285–289.

[12] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, pp. 358–386, 2005.

[13] S.-W. Kim, S. Park, and W. W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases," in *Data Engineering, 2001. Proceedings. 17th International Conference on*. IEEE, 2001, pp. 607–614.

[14] S. Salvador and P. Chan, "Fastdtw: Toward accurate dynamic time warping in linear time and space," in *KDD workshop on mining temporal and sequential data*. Citeseer, 2004.

[15] Y. Sakurai, M. Yoshikawa, and C. Faloutsos, "Ftw: Fast Similarity Search Under the Time Warping Distance," in *Proceedings of the 24th ACM Sigmod-Sigact-Sigart Symposium on Principles of Database Systems*. ACM, 2005, pp. 326–337.

[16] P. Montalto, M. Aliotta, A. Cannata, C. Cassisi, and A. Pulvirenti, "Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining," 2012-09-12.

[17] I. Perfilieva, "Fuzzy transforms," *Transactions on Rough Sets II*, pp. 63–81, 2004.