

# Parallelism in Dynamic Time Warping for Automatic Signature Verification

Y.J. BAE \* and M.C. FAIRHURST

University of Kent at Canterbury, U.K. (Email: mcf@ukc.ac.uk)

## Abstract

*Dynamic Time Warping is a mathematical optimisation technique for sequentially structured problems, which has, over the years, played a major role in providing primary algorithms for automatic signature verification. As this useful method of non-linear, elastic time alignment, still has a high computational complexity due to the repetitive nature of its operations for the optimisation process, this paper proposes a parallel algorithm using a pipeline paradigm, this being chosen with the intention of overcoming possible dead-locks in the highly distributed network. The algorithm was implemented on a transputer network on the Meiko Computing Surface using Occam2, which resulted in a reduction of time complexity by an order of magnitude.*

*Key words: Dynamic Time Warping, elastic time alignment, pipeline paradigm, network efficiency.*

## 1. Introduction

One of the major issues addressed in signature verification systems is how to compare two signature patterns despite their different durations and their non-linear distortion with respect to the time parameter. Dynamic Time Warping (DTW) has, over the years, provided a major tool in overcoming this problem [6, 8, 9, 10]. It has been particularly used to eliminate the timing differences between two differently originating pattern signals by elastically warping one pattern signal [7]. Although this method has been highly successful in automatic signature verification processes, it still has a high computational complexity due to the repetitive nature of its operations for the optimisation process. This has motivated the development of a parallel algorithm for the implementation of the DTW method.

\* Now with SERI, Korea Institute of Science & Technology.

## 2. Basic Concepts

Consider two signals as sequences of feature vectors:

$$\begin{aligned} A &= a_1, a_2, \dots, a_i, \dots, a_I \\ B &= b_1, b_2, \dots, b_j, \dots, b_J \end{aligned} \quad (1)$$

To normalise these two signals with an  $N$ -stage decision process, a sequence of decision functions can be expressed as:

$$D^{(k, x_k)} = \sum_{k=1}^N C_k(q_k, x_k) \quad (2)$$

where  $C_k$  is a contribution function at the  $k^{\text{th}}$  stage for the decision vector  $q_k$  and the state vector  $x_k(a_i, b_j)$ .

DP matching seeks to find the optimum function  $D(k, x_k)$  at the  $k^{\text{th}}$  stage:

$$D^{(k, x_k)} = \underset{q_k}{\text{Optimum}} [D^{(k-1, x_{k-1})} + C_k(q_k, x_k)] \quad (3)$$

Sakoe [7] gave an example of practical implementation of DTW, which is depicted in the diagram of Fig. 1.

## 3. Parallelism for Dynamic Time Warping

A significant potential area of difficulty concerns the fact that the inherent computational complexity of the DTW, when implementation is considered, is very high. In most applications of on-line signature verification, however, there is a real need for fast processing and a rapid verification decision. There is consequently a potentially serious mismatch between the choice of processing algorithm and the practical constraints imposed by the particular application area of interest. This problem can be addressed by considering an implementation strategy which uses the parallel processing paradigm to provide the opportunity for reducing processing times required.

In the diagram of Fig. 1, the calculation for the DP equation is repeated with respect to two variables,  $i$  and  $j$ , which correspond to the number of stages and the size of adjustment window. This iterative operation causes the

main system latency. The proposed parallelism is to resolve the complexity regarding "j".

As can be deduced from the data flow in Fig. 1, DTW is an inherently sequential method in which each stage needs the result of the preceding stage to perform the optimisation at its stage.

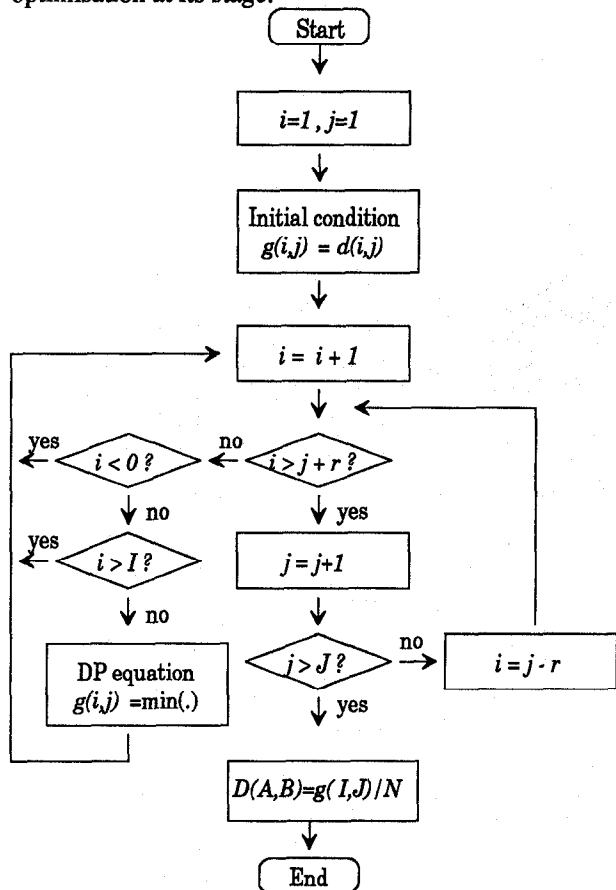


Fig. 1 Sequential DTW implementation

Considering, however, the case of *slope constraint*  $p = 0$  [7], which is widely used for practical applications as, apart from that this provides the fastest implementation, the slope constraint on the warping function has been noted to be generally time-consuming [5], the optimisation process can be depicted as in Fig. 2, which visually indicates the possibility of geometric parallelism: each processing unit performs its own optimisation process within its subsection throughout all the stages while continually communicating boundary data to neighbouring units handling neighbouring subsections. Although the parallel implementation problem appears to be straightforward assuming that a geometric paradigm is adopted, in Occam2 applied in this study, however, communication between processes is realised on a one-to-one, synchronised basis [4]. It is notoriously difficult to debug a system, composed of a large number of local

memory based processors, in which the only communications medium is the set of point-to-point connections between the processors [1]. The pipelined parallel algorithm, therefore, was selected. It has a very simple and straightforward network of one-directional communication links. Fig. 3 describes the algorithm implemented, in which the workers change their positions to maintain a loop network and the master performs an inter-link node.

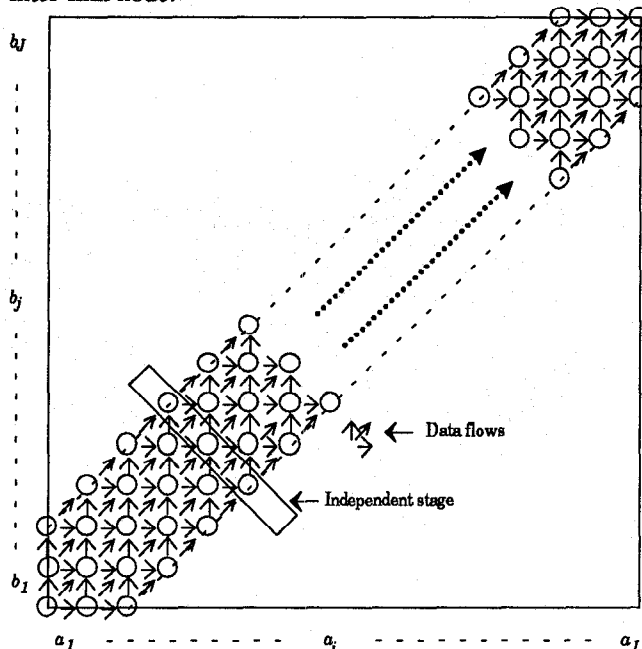


Fig. 2 DTW data flow for scale 0

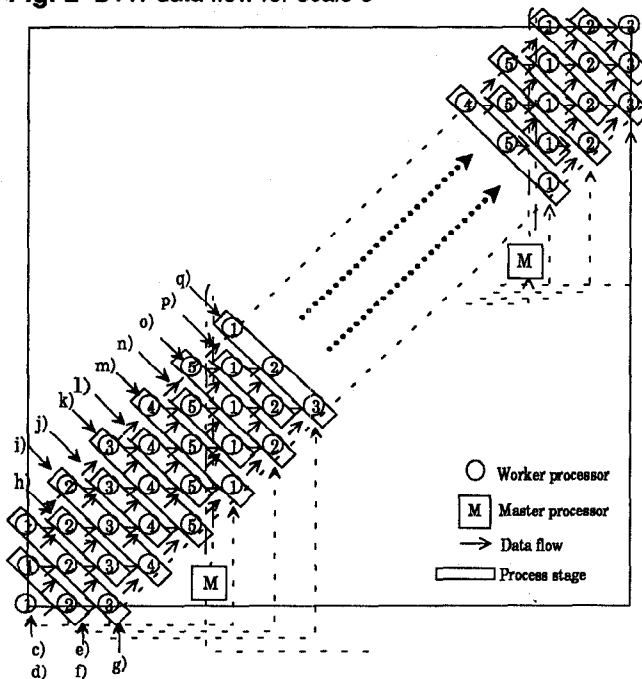


Fig. 3 Parallel DTW

### Configuration

A schematic diagram for the configuration of this algorithm is illustrated in Fig. 4, where the synchronisation and communication are performed on a one-directional communication link.

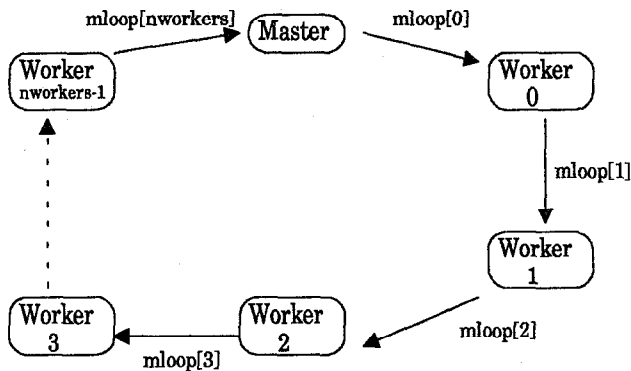


Fig. 4 Network link for the algorithm

### Processes

Each worker is initialised by the master, performs the DTW process for all the stages and sends the result to the master:

```

{{{
PROC worker ( CHAN OF INT in, out)
{{{
SEQ
...initialise --receives two pattern sequences for
PAR DTW from master
count := 0
arg_0 := init_val
atg_1 := init_val
SEQ index = 0 FOR (2 * N) -- iterates DTW job
SEQ
in ? arg_1
IF
arg_2 = init_value
SKIP
TRUE
SEQ
... DTW process --performs DTW
PAR --send DTW result, g(i,j)
out! g_ij
arg_2 := arg_1
arg_0 := g_ij
count := count + 1
IF
(count \ no_workers) = 0 --change
SEQ stage
... change job place

```

```

out! dum --pump the next
TRUE node
SKIP

```

```

}}}
}}}
```

The master initialises the network by pumping the first worker, by-passes the values between workers during the whole DTW processes, and finally receives the result:

### Computational complexity and communication overhead

Since the aim of this parallel implementation is the reduction of computational complexity of the conventional DTW method, and the bottleneck of the conventional method is the repetitive DTW operations for the optimisation process within the range of the adjustment window at each stage, it is sufficient to evaluate the complexity of this algorithm by measuring the number of DTW operations at each stage assuming a time unit which is equivalent to the time to process one DTW operation.

Then computational complexity is expected as:

$$O(N)$$

where  $N$  = the number of DTW stages.

This considerably reduces the complexity of the conventional method:

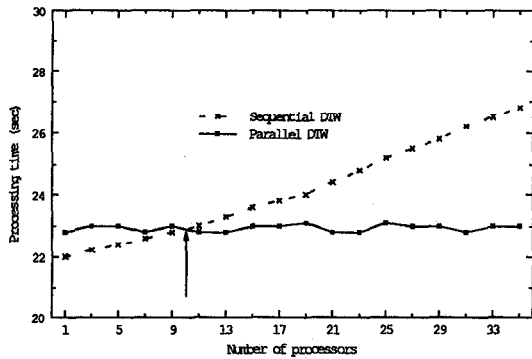
$$O(W * N)$$

where  $W$  = the size of the adjustment window.

The communication overhead is expected to be minor as it needs only four synchronisation cycles per stage, which correspond to two **in** and **out** operations except when the master intercepts one token between the last and the first node, which requires one more cycle. This additional overhead becomes negligible as the number of nodes increases.

## 4. Experimentation and results

The proposed algorithm was successfully implemented using Occam2 and a transputer network on the Meiko Computing Surface [3]. For evaluation, signature samples randomly selected from the signature data base built at the University of Kent at Canterbury for research into image analysis were utilised. Fig. 5 presents the experimental result of the performance evaluation for the pipeline parallelism for DTW of 100 stages, in which the total computation time for 50 iterations of this DTW process is plotted as a function of the number of processors. For comparison, the result using the sequential DTW under the same condition is also plotted.



**Fig. 5** Performance of the Parallel DTW implementation

In Fig. 5, it is noted that real benefit gained from the parallel implementation first appears for eleven T 414 transputers and, accordingly, the magnitude of the benefit compared to the number of processors was small. Considerable communication overheads have been observed: Most of the processing time has been found to be devoted to accessing the input data file, which is stored on the main disk unit in the Sun Sparcstation, not in the transputer network.

It can be seen that the sequential DTW process for the window size of 3 has taken 22.5 seconds on the Meiko network while the same process takes around 5 seconds on a Intel 386DX33 chip based personal computer, which can be regarded as having a similar processing speed to the T414's 10 MIPS. The system latency appears to be over three fourths of the whole processing time and this implies that without this latency, i.e., if implemented in an environment where the communication to the outside world has a reasonable latency (e.g., most globally shared memory), the real benefit can be magnified four times and will appear much earlier.

## 5. Conclusion

A novel approach to exploiting parallelism in the DTW method has been presented. The resulting algorithm, which reduces the computational complexity by an order of magnitude, was successfully implemented and evaluated using Occam2 and a transputer network on the Meiko Computing Surface.

In the experimental results, a considerable overhead due to the file access time was observed. This implies that the parallel implementation of the DTW method, as in many other image-processing applications, needs an efficient architecture which allows for the image data to be accessible to all processors requiring it, with the smallest possible overhead. For this, either globally shared memory architectures or distributed local memory

architectures with an effective interconnection network are thought to be more suitable.

Future work will seek to address the following issues:

- The master may leave the pipeline during DTW processes so as to save two synchronisation cycles per stage which are presently used for the master's processing.
- Parallelism other than the case of *slope constraint*  $p=0$  should be considered.
- An extended approach to VLSI design can be considered, which will require much more work including the consideration of memory complexity.

## References

- [1] E. Barton, "Data concurrency on the Meiko Computing Surface", Int'l Conference on Computer Vision and Display, University of Leeds, 12-15 Jan. 1988.
- [2] Y. Fujimoto et al., "Recognition of handprinted characters by nonlinear elastic matching", Proc. 3rd Int. Joint Conf. on Pattern Recognition, pp. 113-118, Nov. 1976.
- [3] A. J. G. Hey, "Reconfigurable Transputer Network: Practical Concurrent Computation", Scientific Applications of Multiprocessors, Prentice-Hall Pub.
- [4] C. A. R. Hoare, "Communicating Sequential Processes", Prentice Hall Pub., ISBN 0-13-15329-8.
- [5] M. Parizeau and R. Plamondon, "A Comparative Analysis of Regional Correlation, Dynamic Time Warping and Skeletal Tree Matching for Signature Verification", IEEE Trans. on PAMI, Vol.12, No.7, 1990.
- [6] R. Plamondon and G. Lorette, "Automatic Signature Verification and Writer Identification - The State of the Art", Pattern Recognition, Vol.22, No.2, pp.107, 1989.
- [7] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word recognition", IEEE Trans. on ASCP, Vol.26, No.1, 1978.
- [8] Y. Sato and K. Kogure, "On-Line Signature Verification based on Shape, Motion and Handwriting Pressure", Proc. 6th Int. Conf. on Pattern Recognition, Vol.2, pp.823-826, München, 1982.
- [9] M. Yasuhara and M. Oka, "Signature Verification Experiment based on Nonlinear Time Alignment : A Feasibility Study", IEEE Trans. on SMC, Vol.17, pp.212-216, 1977.
- [10] M. Yoshimura, Y. Kato, S. Matsuda and I. Yashimura, "On-line Signature Verification Incorporating the Direction of Pen Movement", IEICE Trans., Vol.E74, No.7, pp.2083-2091, 1991.