Elastic matching in linear time and constant space

Scott MacLean
University of Waterloo
200 University Ave. West
Waterloo, Ontario, Canada
smaclean@uwaterloo.ca

George Labahn
University of Waterloo
200 University Ave. West
Waterloo, Ontario, Canada
glabahn@uwaterloo.ca

ABSTRACT
Dynamic time warping (DTW) is well known as an effective method for model-based symbol recognition. Unfortunately, its complexity is quadratic in the number of points present in the symbols to be matched. In this paper, we propose a greedy approximate solution to Tappert's dynamic program formulation of DTW, and show empirically that it performs as well as the exact solution while requiring only linear time to compute.

Categories and Subject Descriptors
I.5 [**Pattern Recognition**]: Applications—*Text processing*

General Terms
Algorithms, Experimentation, Performance

Keywords
Symbol recognition, Dynamic time warping, Elastic matching

1. INTRODUCTION
Dynamic time warping (DTW) is a signal processing technique providing a measure of the difference between two time series. It was extended by Tappert in 1982 to two dimensional data points for use in symbol recognition as a type of elastic matching [7]. Over time, this approach and its variants have proved to give an accurate measurement of the difference between two digital ink strokes. This paper presents a new, approximate version of DTW using a greedy approach.

In the most basic form of DTW, as applied to symbol recognition, one is given a *model* stroke and an *input* stroke, and is asked to measure the distance between them. A large distance corresponds to highly dissimilar strokes, while a small distance indicates that the strokes trace out similar curves. To compute the distance, each point in the input stroke is matched with a point in the model stroke and a distance measure is applied to each matched pair. The distance between the strokes is given by the sum of the distances between matched pairs. The goal is to match up the points in a way that minimizes the total distance.

By introducing additional constraints, Tappert solved this optimization problem using a dynamic program. In his formulation, a digital ink stroke is treated as a pair of simultaneous time series $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ representing the x- and y-coordinates of the points sampled by the tablet digitizer. Let x, y be coordinate series of n points representing the input stroke, and $\hat{x}, \hat{y}$ be similar (of m points) for the model stroke. Tappert's contraints are as follows:

1. The first input point is matched to the first model point;
2. The last input point is matched to the last model point; and
3. If the ith input point is matched to the jth model point, then the $i+1$st input point is matched to the jth, $j+1$st, or $j+2$nd model point.

Given a distance function $d(i, j)$ between the points $(\hat{x}_i, \hat{y}_i)$ and (x_j, y_j), these constraints induce the dynamic program $D[i, j]$, the minimal distance between model points up to the ith and input points up to the jth, as follows:

$$\begin{aligned} D[1,j] &= d(1,j) + D[1,j-1] \\ D[2,j] &= d(2,j) + \min\{D[2,j-1], D[1,j-1]\} \\ D[i,j] &= d(i,j) + \min\{D[i-k,j-1] : k = 0,1,2\} \end{aligned}$$

Note that many distance functions d are possible. For cursive writing, Tappert uses

$$d(i,j) = \min\left\{\left|\hat{\theta}_i - \theta_j\right|, \left|360 - \left(\hat{\theta}_i - \theta_j\right)\right|\right\} + |\hat{y}_i - y_j|,$$

where θ_i is the tangent angle at (x_i, y_i). We have found that including $|\hat{x}_i - x_j|$ improves results for our own work with mathematical symbol recognition. (In any case, all terms should be normalized so as to have equal weight.)

This basic approach can be extended in various ways. For example, Scattolin proposed to use weights to increase the influence of those points in the model stroke which define its characteristic shape [6]. In this paper, we are not concerned

DAS '10, June 9-11, 2010, Boston, MA, USA

with such extensions, but rather with the complexity of the basic technique itself.

Each table cell requires only constant time to compute, but there are $\mathcal{O}(nm) \approx \mathcal{O}(n^2)$ table cells to be computed and stored. In situations such as mathematical expression recognition, symbol libraries may be quite large, the strokes themselves may be long and complex (ξ, for instance), and symbol recognition may be invoked several times if there are several ways to partition a large input into distinct symbols. Quadratic matching time per stroke can therefore consume a significant proportion of total processing time in the context of a larger system. For example, in the MathBrush math recognition system, real-time recognition feedback is required [2]. The quadratic cost of DTW was found to be unacceptable in this case, prompting our development of a faster variant.

The next section of this paper presents a greedy approximation algorithm to the dynamic program given above. Following the algorithm, we describe an empirical study demonstrating the technique's performance, contrast our proposed approach with some related work, and offer some conclusions on the algorithm's applicability.

2. GREEDY DYNAMIC TIME WARPING

We motivate our algorithm by some straightforward observations about Tappert's constraints (reproduced in the previous section). Let $I_1, I_2, \ldots, I_n$ be the points comprising the input stroke, and $M_1, M_2, \ldots, M_m$ be similar for the model stroke. According to constraints 1 and 2, I_1 must be matched to M_1 and I_n must be matched to M_m. By constraint 3, I_2 must be matched to one of M_1, M_2, M_3, and I_{n-1} must be matched to one of M_m, M_{m-1}, M_{m-2}. Similarly, supposing I_i is matched to $M_{f(i)}$, I_{i+1} must be matched to one of $M_{f(i)}, M_{f(i)+1}, M_{f(i)+2}$, and I_{i-1} must be matched to one of $M_{f(i)}, M_{f(i)-1}, M_{f(i)-2}$.

Tappert's dynamic program finds a globally-optimal matching satisfying these constraints. Our approximate version is to simply match endpoints to endpoints, then to greedily choose the locally-optimal matching from the available options for each intermediate point along the input stroke. To ensure endpoints are matched together, we perform a two-sided match beginning at the start and end of the strokes and working simultaneously toward the middle.

There are two potential problems we must be aware of in this scheme, particularly if the number of points differs significantly between the input and model strokes. After matching all the input points to model points, there may be a large number of points in the middle of the model stroke which were never considered by the algorithm. Conversely, the algorithm may run out of model points available for matching before all of the input points have been considered. These situations are exemplified schematically by Figures 1 and 2, respectively. In the figures, dashed lines indicate pairs of matched points, and grey points are unmatched and problematic.

To account for these cases, we include the following two rules in our procedure:

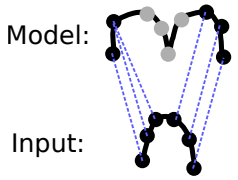

Figure 1: Many unmatched model points remain after matching every input point.

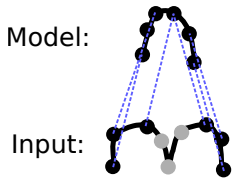

Figure 2: No more model points are available, but several input points remain to be matched.

1. After matching each input point to a model point, implicitly match the center-most input point to every second model point not yet considered for matching. (This process simulates skipping over model points, as permitted by Tappert's constraints.)
2. If there are no available model points to consider for matching, match all remaining input points to the center-most model point.

These rules immediately give an algorithm for approximate dynamic time warping, listed in Algorithm 1.

Regardless of the length of the strokes, the algorithm uses a fixed number of variables to track point indices, local and global match costs, and which local match choice was optimal. In each iteration of the main while loop (line 7), f_I is incremented and b_I is decremented, so only $n/2$ iterations are possible. Notice that the loop body requires only constant time, assuming d requires constant time. If the else clause at line 24 is invoked, then the loop at line 30 will not be entered; otherwise that loop will run at most $m/2$ times. The algorithm's runtime is thus linear in the number of input and model points.

This algorithm is suitable for an efficient practical implementation. If the x- and y-coordinates comprising a stroke are stored in separate arrays, then it is plausible that, after appropriate subdivision, a model stroke and an input stroke may be contained in a few cache lines. Branch prediction for each of the if statements in the main loop should only fail once each. From a practical perspective, the most expensive lines are the argmin operations, which must find the least of three values.

3. EVALUATION

To evaluate its performance, we compared our proposed greedy dynamic time warping algorithm to an implementation of Tappert's original formulation. All single-stroke symbols were extracted from a publicly available, ground-truthed corpus of handwritten mathematical expressions ([3]), giving 70 distinct symbol classes of various sizes. The cor-

Algorithm 1 Greedy approximate dynamic time warping.

Require: Input and model strokes of n, m points respectively; distance function $d(i, j)$ as in the previous section.

```
// Initialize indices to the start and end of strokes
f_I ← 1; b_I ← n; f_M ← 1; b_M ← m
// Match endpoints
c_f0 ← d(f_M, f_I); c_b0 ← d(b_M, b_I)
c ← c_f0 + c_b0
f_I ← f_I + 1; b_I ← b_I − 1
while f_I < b_I do
  // Measure relevant local match costs
  r ← b_M − f_M
  if r > 0 then
    c_f0 ← d(f_M, f_I); c_b0 ← d(b_M, b_I)
    c_f1 ← d(f_M + 1, f_I); c_b1 ← d(b_M − 1, b_I)
    if r > 1 then
      c_f2 ← d(f_M + 2, f_I); c_b2 ← d(b_M − 2, b_I)
    else
      c_f2 ← ∞; c_b2 ← ∞
    // Choose minimum-cost match locally
    i ← argmin {c_fk : k = 0, 1, 2}
    j ← argmin {c_bk : k = 0, 1, 2}
    c ← c + c_fi + c_bj
    // Advance to the next points under consideration
    f_M ← f_M + i; b_M ← b_M − j
    f_I ← f_I + 1; b_I ← b_I − 1
  else
    // Model exhausted; match remaining input points to last matched point
    while f_I < b_I do
      c ← c + d(f_M, f_I)
      f_I ← f_I + 1
// Input exhausted; match remaining model points to last matched point
while f_M < b_M do
  c ← c + d(f_M, f_I)
  f_M ← f_M + 2
return c
```

pus contains handwriting samples from 20 writers. We aggregated the first k instances of each symbol written by each writer together into a symbol library, for $k = 1, 2, 3, 4, 5$, and tested the algorithms on the remaining symbols. (If fewer than k instances were available for a particular writer, then we used all instances but one.) Using both the original dynamic time warping algorithm and our greedy variant, each input symbol was compared against all library symbols. The class of the library symbol with lowest match cost (i.e. the nearest neighbour) was declared the winner and compared to the actual class of the input symbol. We used Manhattan distance for the distance function d.

We also varied the number of stroke points shown to the algorithms. Strokes were either left unprocessed or subdivided into one of 6, 12, 18, 24, 30, 36, or $\sqrt{n}$ points (n being the number of points in the original stroke). We used an arclength subdivision algorithm that gives preference to points of high curvature.

The results of our tests are summarized by the graphs shown in Figures 3 through 5. The first graph shows how the algorithms' performance varies with a fixed-size symbol library but with strokes subdivided into varying numbers of points. The other graphs show the algorithms' performance using a fixed subdivision strategy, but varying the number of example symbols in the symbol library.

In all graphs, the bars show the cumulative time taken by the algorithms. The number of input symbols and DTW invocations for each library size is summarized by Table 1. The lines show recognition accuracy: dashed lines indicate the standard DTW algorithm and solid lines indicate our variant. The lines marked "Correct" measure the proportion of input symbols whose class matched that of the library symbol with lowest match cost. The lines marked "Top 5" measure the proportion of input symbols whose class appeared within the five lowest-cost matched symbols.

# Symbols per user	# Input symbols	# Stroke Cmps
1	14667	16559043
2	13625	29579875
3	12674	39568228
4	11826	46949220
5	11078	52266004

Table 1: Number of tests during our experiments.

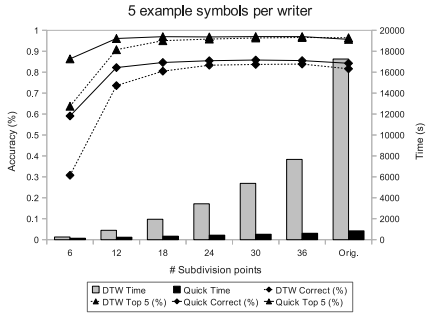

Figure 3: Test results using five library symbols per class per writer.

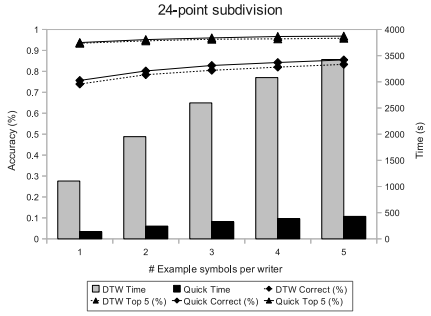

Figure 4: Test results using subdivision into 24 points.

In our experiments, the greedy variant of DTW was always at least as accurate as and much faster than the original algorithm. It is particularly accurate when the strokes have few points, as its accuracy climbs quickly as the number of points increases before leveling off. The standard algorithm

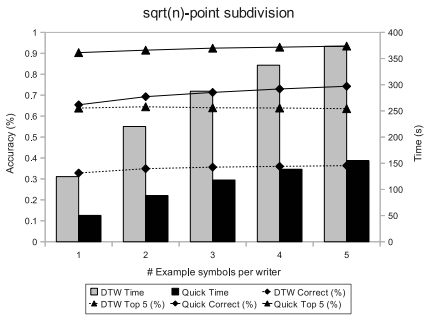

Figure 5: Test results using subdivision into $\sqrt{n}$ points.

requires relatively many points before it reaches maximum recognition accuracy.

Intuitively, with fewer points per stroke, there are fewer possibilities for the greedy algorithm to "go wrong" compared to when there are many points in a stroke. Indeed, the accuracy of the greedy algorithm drops slightly after 36 points per stroke. However, it is always competitive with, and most often superior to, the accuracy of standard DTW.

4. RELATED WORK

There are several other techniques for approximating the DTW distance measure in linear time. Salvador and Chan's FastDTW algorithm uses a multilevel coarsening/refining process and obtains 1-D distance measurements within about 2% of the quadratic DTW algorithm using linear time and space [5]. The method presented in this paper has much larger mean error, on the order of 100%, but different goals. FastDTW is intended for use as a distance measure, whereas we are only interested in the properties of the greedy algorithm in symbol recognition applications. Our experiments indicate that, despite its large discrepancy from the optimal solution, the greedy algorithm is still able to match inputs to the corresponding models at least as well as DTW.

Ratanamahatana and Keogh pointed out that, using banding constraints and lower-bounding techniques, the DTW algorithm needs to compute only $\mathcal{O}(n)$ of the table cells $D[i, j]$ in an amortized sense when matching an input to a large collection of models. They suggest that there is no sense in searching for other linear time variants [4]. Our greedy approach can be seen as a banding constraint that obviates the dynamic programming table altogether, and always requires $\mathcal{O}(n)$ time. While banding and lower-bound checks are useful in general, for some applications it may be more appropriate to choose a particularly effective band in which it is fast to compute the dynamic time warping distance.

Recently, Golubitsky and Watt proposed a linear-time symbol recognition algorithm based on comparing the first d coefficients of truncated Legendre-Sobolev series representing the ink strokes [1]. Their algorithm has slightly poorer recognition performance than DTW, but can be used in a streaming context, processing each stroke as it is sampled by the digitizer using $\mathcal{O}(1)$ operations per point and $\mathcal{O}(d)$ further operations upon completion of the stroke. Such streaming usage is not supported by our greedy algorithm as it matches strokes beginning simultaneously at both endpoints.

5. CONCLUSIONS

In this paper, we have presented a straightforward greedy approximation to the dynamic time warping algorithm, with particular attention to the application of symbol recognition. The algorithm runs in linear time and uses only a constant amount of memory. Our experiments showed that it is at least as accurate as DTW for short time series (an appropriate limitation for most symbol recognition applications), and much faster. The experiments were performed on a realistic data set of 15796 handwritten symbols of 70 distinct classes, written by 20 different writers.

In the case of symbol recognition, the precise value given by a distance measure between input and model strokes is not as important as the relation between several distances. So long as the correct model stroke gives the lowest distance measure, the application will perform well. It is worthwhile to investigate the extent to which this observation applies to other application areas. Another area requiring further investigation is the behaviour of our greedy variant using different pointwise distance functions d.

In practice, the processor time saved by using such a fast distance measure can be used to boost recognizer performance either by including several complementary fast distance measures and combining their results, or by using context-sensitive processing and domain models (e.g. grammars, dictionaries, etc.). The greedy approximation algorithm described in this paper has essentially made feasible a hybrid model for math recognition using two-dimensional parsing with multiple symbol grouping possibilities [2].

6. REFERENCES

[1] O. Golubitsky and S. Watt. Computation of similarity between handwritten characters. In *Proc. Document Recognition and Retrieval XVI*, pages C1–C10, 2009.

[2] G. Labahn, E. Lank, S. MacLean, M. Marzouk, and D. Tausky. Mathbrush: A system for doing math on pen-based devices. In *Proc. of the Eighth IAPR Workshop on Document Analysis Systems*, 2008.

[3] S. MacLean, D. Tausky, G. Labahn, E. Lank, and M. Marzouk. Tools for the efficient generation of hand-drawn corpora based on context-free grammars. In *Proc. of the Sixth Symposium on Sketch-Based Interfaces and Modeling*, 2009.

[4] C. A. Ratanamahatana and E. J. Keogh. Three myths about dynamic time warping data mining. In *Proc. of the 2005 SIAM International Conference on Data Mining*, pages 506–510, 2005.

[5] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.*, 11(5):561–580, 2007.

[6] P. Scattolin. Recognition of handwritten numerals using elastic matching. Master's thesis, University of Concoria, Montreal, Canada, 1995.

[7] C. C. Tappert. Cursive script recognition by elastic matching. *IBM J. Res. Dev.*, 26(6):765–771, 1982.