Dynamic Time Warping for Pure Data

Pedro Lopes Joaquim A. Jorge
Department of Information Systems and Computer Science
INESC-ID/IST/Technical University of Lisbon
R. Alves Redol, 9, 1000-029 Lisboa
Portugal
{pedro.lopes, jorgej}@ist.utl.pt

ABSTRACT

We present an implementation of the Dynamic Time Warping algorithm for the Pure Data programming environment. This algorithm is fairly popular in several contexts, ranging from speech processing to pattern detection, mainly because it allows to compare and recognize data sets that may vary non-linearly in time. Our contribution is easily portable to a wide number of platforms, where Pure Data is available. Throughout this document we describe relevant work that inspired our proposal and present the core concepts of our implementation.

We conclude with an evaluation of our Dynamic Time Warping implementation in two perspectives: performance and adequacy towards gesture recognition. The performance tests suggest that it is suited for realtime contexts, where algorithmic efficiency is of utmost importance. Finally we present a case study where our implementation was used successfully to accommodate gesture recognition on an existing application.

Keywords

Pure Data, Dynamic Time Warping, Gestural Recognition

1. INTRODUCTION

Dynamic Time Warping (DTW) is a very popular algorithm, it has been around for a long time and actively used in numerous fields of research, from Speech Recognition [14, 9] up to MIDI-Audio track alignment [13]. This research strives to fill a gap within the Pure Data external repository and implement a DTW-capable external that is easy to operate by end users, allowing them to create new possibilities in pattern-recognition techniques such as gesture, image or audio processing

Including this section, this document is organized into six sections. On the next section will present the needed background to briefly explain DTW and overview some implementations for the Max/MSP environment, closely related to Pure Data (PD) [8]. For the third section we present our solution, highlighting the current state of the implementation. For the fourth, we present the performance analysis of the current version of the external and point out to a multimodal application already benefiting from it towards gesture recognition. Finn ally on the fifth and sixth sections, we draw conclusions and build upon the future work on this research.

PdCon'11, 8 August–10 August 2011, Weimar, Germany.

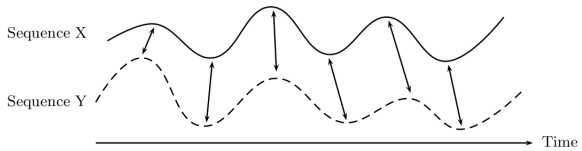

Figure 1: The optimal alignment of two time-dependent series, where arrows denote the aligned positions. [6]

2. BACKGROUND AND RELATED WORK

We now present an explanation of DTW, alongside an example in a Pure Data user perspective. Also we present relevant work in DTW implementations, optimizations and an overview of externals written for Max/MSP that can serve as inspiration for our research.

2.1 DTW Algorithm

In rough terms, DTW is a method that measures the similarity between two given sequences (e.g. time series, amplitude data, and so forth), the algorithm searches the optimal alignment between those two series, as depicted in Figure 1. Thus these sequences are said to be "warped" non-linearly in time in order to determine the optimal match. Hence the algorithm is defined as independent of non-linear variations in the time dimension [9].

A more meaningful example can be given: detect a red t-shirt passing in front of a camera. It is common sense that the motion can have de/accelerations, thus if you want it to be matched against a stored one (a couple of seconds of previously recorded film) one must account for these non-linear variations in time. The DTW gives a measure of similarity between two series, hence the output of the algorithm processing is a "distance" value, of how far (or conversely, how close) the sequences are. When dealing with gesture recognition the DTW will determine how correct the sample input data is when compared to one template from your database of gestures [2].

2.2 DTW Implementations

We focus on open-source implementations of the DTW algorithm. Since our target is the PD environment, we present two different C and C++ implementations. The first DTW is written in C [11], thus easily portable to PD. It is multidimensional (imagine using it to compare pixels, hence two dimensions) and reads data from textfile input. The data series are written in consecutive lines, and for each line it can accommodate as many variables as the defined dimensions. Due to the simplicity of the approach of the authors, this was chosen and later adapted into the external.

A more complex implementation resides within the lbimproved project[1], this project is harder to adapt since it is focused on providing other functionalities and depends on external libraries.

[1] `http://code.google.com/p/lbimproved/`, last access on 7/1/2010

2.3 Max/MSP implementations

Two DTW externals are available for Max/MSP environment, one is publicly available, while the second is a result of an academic research.

The first implementation is available for Max/MSP users, it performs DTW and reads data accordingly to FTM definition [10]. Although many Max externals are easy to interconnect with PD, this resides on the availability of the code, which for the case is not open. Hence even regarding the recent efforts in porting FTM to PD [3], cannot be considered.

Secondly another DTW implementation was described in [12], it goes one step further by addressing result-space partitioning in a more efficient manner than the previously cited works. Although the authors intended to port this research to PD, at the time of writing, the effort had been put off[2].

3. PROPOSED SOLUTION

As seen in the previous section, there has been a lack of DTW in Pure Data, hence this effort has been carried out to create an easy to use implementation. We will present the guidelines for our solution, first describing the desired implementation, and finally, the current state of the DTW external.

3.1 Conceptual Implementation

3.1.1 Database Templates

The DTW must feed from some database of templates in order to perform comparison against a given input. This templates are defined in table 1. In a patch-situation a template can be viewed a gesture recorded for later analysis, or an image that should be matched by a video stream and so forth.

Template Parameter	Meaning	Data type (C)
Data	A series of values that will be used for comparison with incoming data	*n*-dimensional float array
Name	name to be reported to the output if this series is matched	string[4]
Threshold	value for which this series is acceptable. [5]	float
ID	unique identifier [6]	int
Origin	“file.templates" or “runtime"	string[4]

Table 1: Definition of the Template concept.

[4]String is not an actual type in C, thus here it stands for a char array with a maximum of 256 characters.
[5] The Threshold can be set manually, or automatically collected from database if the first is not set.
[6] Sequential IDs ensure access to all templates and uniqueness.

The ideas are analogous to Miller’s Bonk~ [8], being heavily message dependent instead of creation arguments. However if any arguments need to be passed to the compiled external at load time, they will have to be explicitly written in argument box of the visual object (or code correspondent in PD’s fileformat).

3.1.2 Interaction overview

The external is implemented in a straight forward fashion based on [5], the goal is that the external should be easy to use, hence have a understandable message communication, as depicted in Figure 2.

[2]The authors of this work [12] were contacted in May 6, 2010 and replied that the Pure Data port is not being considered.

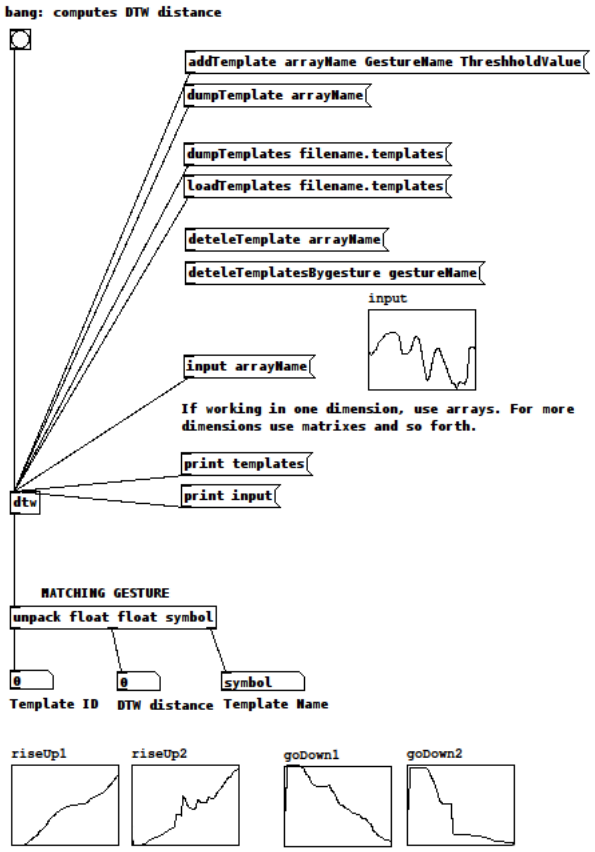

Figure 2: Mock up of desired help file and messages for controlling the DTW external.

3.1.3 Find-best-match approaches

When the user “bangs" to activate the DTW comparison of the input data (i.e.: Figure 3) against all templates, the DTW distance value is computed for each pair input-template[*i*]. *i* being the number of database stored templates. Running every pair will yield a best guarantee that the best candidate is selected, as depicted in Figure 4, thus the match will be more likely to be correct at the user level.

However, optimization is possible. By improving upon result space partitioning we can, for instance, try one template of each “name" and if the it does not match move to the next “name" on the list. The situation is described in Figure 4, the template is matched quickly since the algorithm uses a different result partitioning, this can ultimately result in wrong matching under very specific circumstances, if a latter sample would reveal to be a better match than the current selected one[3]. In [12] a post-processing stage is added, using a N-best strategy (N = 3), that is, displaying continuously the three best matched gestures. Similar approaches can be used to make recognition faster, in the PD implementation.

3.1.4 Obtaining thresholds

[3]This depends heavily on the quality and intent of the stored data within the templates, if all templates under a target “name" are much different, this wrong-match will not happen. Conversely if the user is analysing very similar data, with poorly defined thresholds

d

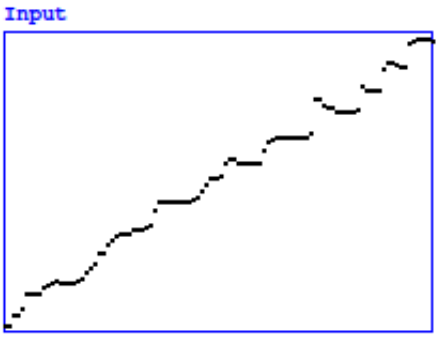

Figure 3: Input frame to be matched (blue).

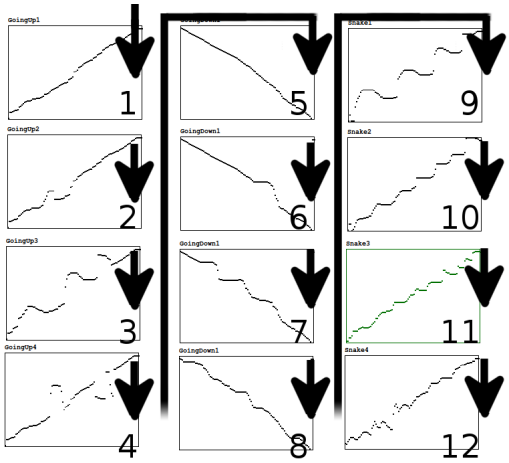

Figure 4: By comparing all templates we can always find the optimal match (green) to the input frames (blue).

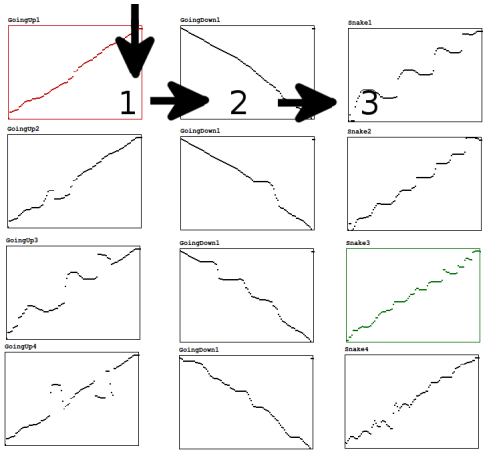

Figure 5: "Shortcut like" optimizations can result in poor matching (red) of the input (blue), because the optimal (green) was not scanned.

After storing any number of templates under a "name", to compute a threshold automatically the user can train the system. Training the threshold is done by computing the average difference of all templates under the same name. However, this is not optimal approach, as newer techniques have to be researched.

3.1.5 Data Handing

Data fed to the templates should use an understandable and efficient format. Iemlib provides a suitable matrix implementation [7], while PD's internal array serves for one dimensional purposes. For n-dimensionality a solution is still to investigate. Textfiles are also relevant for persistent data, thus the original format proposed by [11] is used. Concluding, regarding data handling, the external has to be able to load data from textfile (n-dimensional), iemmatrix (2 dimensional) and array (1 dimensional).

3.1.6 Optimization

This work is not focused on algorithmic optimizations, although that are relevant an possible. The external calls a isolated function for DTW computation, this can be replaced or pointer-function easily. This can switch, for example, the DTW to parallel - faster on nowaday's multicore computers.

3.2 Concluded Implementation

The DTW class performs the basic required functions, and obviously computes the distance between two given *n*-dimensional series. The data handling is still textfile based, thus both templates and input are read from disk in the aforementioned format. Templates are loaded at start and stored in memory, so currently they are not dynamic. The template format described in Table 1 is implemented in the class. The input is loaded from file on each user request ("bang" object to compute DTW distance) so it suffers from I/O operations to read the data to memory. All these aspects will, of course, be corrected as the implementation iterations move forward.

Also currently a Parallel version of the external is implemented, using a custom-compiled Pure Data with OpenMP support and the DTW class is parallelized to account for multicore distribution of matrix computations load. This will ultimately result in an evaluation of parallelizing techniques for Pure Data environment, similar to what has been done in FAUST [15].

Regarding result space optimization and automatic thresholding, it is not yet concluded, thus thresholds are user defined for each template and in the best-match all templates are analysed against.

4. RESULTS AND DISCUSSION

Although not aiming for optimization of the DTW implementation at this stage, any meaningful contribution would not be complete without an analysis of the current performance of the DTW class in PD. Besides that we present a working example that already benefited form the DTW external in order to perform gestural recognition.

4.1 Performance Analysis

For the performance test we used databased of amplitude samples, recorded directly form a microphone into several arrays. Each template represents a gesture, such as "tapping on the microphone one time" or "dragging the finger on the capsule". For each template four gestures were recorded and stored in the arrays that are loaded into the DTW via textfiles. The DTW algorithm for a 1-dimensional series can be solved in polynomial time[2], which is the case for our sampled amplitude signal.

Table 2 shows the system used for performance test. We used an eight sample database, with arrays of 100 samples recorded from microphone input. The average run of the full DTW cycle takes $\approx$0.010255868 seconds to compare all data, compute DTW distance and thresholds (this accounts for I/O read time of input file).

Type	Specifications
Hardware	Intel Core 2 Duo P9500 @ 2.53GHz
OS	Ubuntu Maverick 10.10 Kernel Linux 2.6.35-23-generic-pae
Pure Data	PD Vanilla 0.43 (Test Release) with OpenMP enabled

Table 2: Test system specifications.

This performance result, as will overview in the next section, is enough for stable realtime gesture recognition.

4.2 Working prototypes with DTW external

We describe a project that used our Pure Data DTW external successfully to accomplish gesture recognition. In [4] the user controls a DJ-like application with novel feet gestures, as depicted in Figure 6. While DTW has long been used for many years to recognize speech [9], its application to gesture recognition is relatively recent [2]. A low-cost microphone captures the friction of the user's feet on the floor; this sample window is analysed with DTW in order to identify gestures in a time-independent and scalable manner. The recognized gesture is shipped from PD to the target DJ application (via Open Sound Control over TCP socket) that will process and execute the command.

The user gestures are captured with a microphone under the surface (of his feet). The amplitude of the sound is sampled in a frame of 100 samples of a running 100Hz sample rate (which later can be increased to the desired dimension for optimal results), the frame is cross compared with the records stored in the database (training data previously fed). The comparison uses the DTW algorithm in a way that similar gestures can be compared with time independence (a gesture performed slower/faster than the templates is also recognized) and noise can be filtered out too. Also we must stress that other possibilities exist towards accurate gesture mapping, such as [1], although the emphasis of our research is addressing DTW implementation for the Pure Data environment.

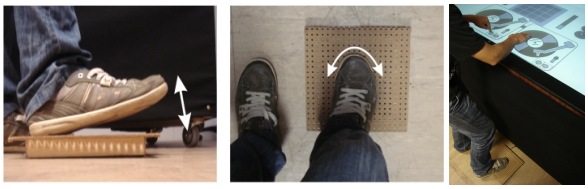

Figure 6: Augmented DJ application with feet gestures recognized by audio DTW analysis [4].

The training data is comprised of 8 template gestures, four samples for the gesture "Kick" and the remaining for the gesture "Drag", the sample data is shown below. Each gesture triggers a DJ action in the multitouch tabletop: kick sets the tempo, and drag turns the metronome action of and off, as depicted in Figure 6.

5. CONCLUSIONS

We described the implementation of a Pure Data environment trainable-DTW external, suited for comparing multi-dimensional data, in a efficient manner. The progress was reported and the desired architectural concepts, defined and discussed. We concluded with a performance analysis of the algorithm efficiency and pointed a project already built upon this DTW implementation.

6. FUTURE WORK

As noted throughout this document several steps are yet to be established before the DTW external can be available for the large spectra of the Pure Data users, from these we highlight: loading data directly from some matrix implementation (e.g: iemmatrix) and/or from array data; changing DTW on the fly (different implementations, parallel, optimization approaches, and so forth); and finally define a message format for the output (e.g.: list of atoms: first item gesture name or number and second distance value).

7. ACKNOWLEDGEMENTS

This research was supported in part by FCT through research grant MIVIS PTDC/EIA/104031/2008 which we gratefully acknowledge. We thank Guilherme Fernandes and the Pure Data developers/community whose help was also a part of this upcoming external.

8. REFERENCES

[1] A. Cont, T. Coduys, and C. Henry. Real-time gesture mapping in pd environment using neural networks. In *Proceedings of the 2004 conference on New interfaces for musical expression*, NIME '04, pages 39–42, Singapore, Singapore, 2004. National University of Singapore.

[2] C. Harrison and S. E. Hudson. Scratch input: creating large, inexpensive, unpowered and mobile finger input surfaces. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 205–208, New York, NY, USA, 2008. ACM.

[3] T. M. IOhannes m zmölnig and W. Ritsch. Freer than max - porting ftm to pure data. In *Proceedings of The LAC 2008 - International Linux Audio Conference*, 2008.

[4] P. Lopes, G. Fernandes, and J. A. Jorge. Trainable dtw-based classifier for recognizing feet-gestures. In *RECPAD 2010: Proceedings of the 16th Portuguese Conference on Pattern Recognition*, 2010.

[5] I. m zmölnig. Howto write an external for puredata. Technical report, institut for electronic music and acoustics, 0.

[6] M. Müller. *Information Retrieval for Music and Motion*, volume XVI. Springer, 2007.

[7] T. Musil. Iemlib for pd. Technical report, IEM Gratz, 2003.

[8] A. T. Puckette M. and Zicarell. Real-time audio analysis tools for pd and msp. In *Proceedings, International Computer Music Conference (ICMC). San Francisco: International Computer Music Association*, pages pp. 109–112, 1998.

[9] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:43–49, 1978.

[10] N. Schnell, R. Borghesi, D. Schwarz, F. Bevilacqua, R. Muller, and I. C. Pompidou. Ftm - complex data structures for max. In *In Proc. ICMC*, 2005.

[11] A. Slater and J. Coleman. Dtw implementation in ansi-c. Phonetics Laboratory, Faculty of Linguistics, Philology and Phonetics, 0. `http://www.phon.ox.ac.uk/files/slp/Extras/dtw.html`, last accessed on 7/1/2010.

[12] F. Todoroff T., Bettens. Real-time dtw-based gesture recognition external object for max/msp and puredata. In *SMC*, 2009.

[13] R. J. Turetsky and D. P. W. Ellis. Ground-truth transcriptions of real music from force-aligned midi syntheses. In *ISMIR*, 2003.

[14] P. West. The extent of coarticulation of english liquids: An acoustic and articulatory study. In *Proceedings of the XIVth ICPhs99 International Congress of. Phonetic Sciences*, 2009.

[15] D. F. Yann Orlarey, Stephane Letz. Parallelization of audio applications with faust. In *Proceedings of the SMC 2009 - 6th Sound and Music Computing Conference, 23-25 July 2009, Porto - Portugal*, 2009.