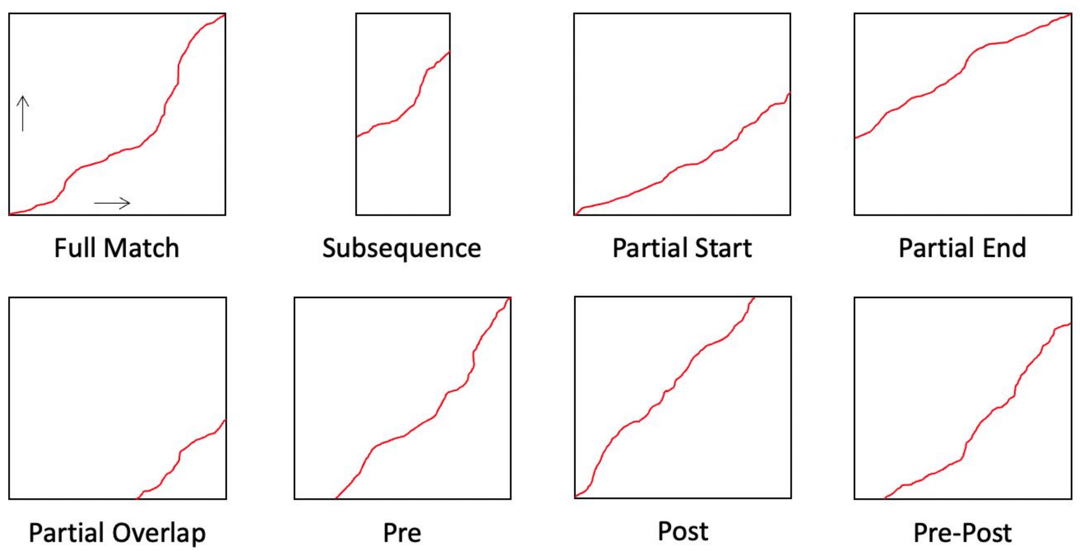
FlexDTW: Dynamic Time Warping With Flexible Boundary Conditions

Irmak Bükey[1] Jason Zhang[2] TJ Tsai[3]
[1] Pomona College [2] University of Michigan [3] Harvey Mudd College

Problem Statement

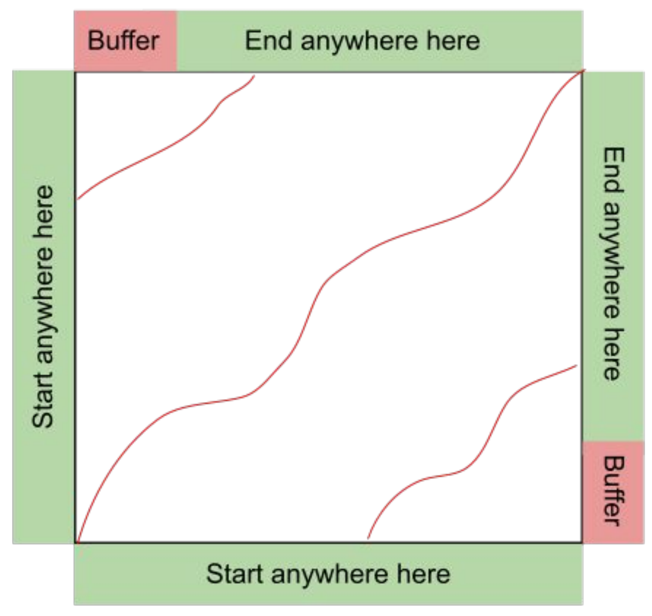
- A standard tool for aligning audio performances is dynamic time warping (DTW). DTW finds the lowest cost alignment path through a pairwise cost matrix.
- DTW and its variants have specific assumptions about the boundary conditions of the alignment path. DTW assumes the alignment path starts in one corner of the cost matrix and ends in the opposite corner. Subsequence DTW assumes that the alignment path starts on the longer edge of the cost matrix and ends on the opposite edge.
- In practice, the boundary conditions may not satisfy these assumptions or may not be known in advance. For example, when aligning Youtube performances of classical music, boundary conditions may be affected by silence or applause at the beginning or end, or one video having a different number of movements than another video.

Our goal is to develop an alignment algorithm that can flexibly handle a wide variety of boundary conditions.



FlexDTW: Overview

- FlexDTW allows the alignment path to start anywhere on the left or bottom edge, and to end anywhere on the right or top edge. A short buffer region is imposed to avoid short, degenerate alignment paths near the top left and bottom right corners.
- To fairly compare alignment paths of very different length, we must use a path cost measure that normalizes by the path length. We could backtrack from every position (i,j) to determine the length of the alignment path ending at (i,j), but this would require a prohibitive amount of additional computation
- The **key insight of FlexDTW** is that Manhattan distance can be computed by simply knowing the starting point of the alignment path (not the actual path itself). This information can be computed recursively during dynamic programming, eliminating the need for backtracking.



FlexDTW: Algorithm

1. Initialize
   - Cumulative cost matrix $D \in \mathbb{R}^{N\times M}$: $D[i,0] = C[i,0], \quad i = 0,\ldots,N-1$; $D[0,j] = C[0,j], \quad j = 0,\ldots,M-1$
   - Backtrace matrix $B \in \mathbb{Z}^{N\times M}$
   - Starting point matrix $S \in \mathbb{Z}^{N\times M}$: $S[i,0] = -i, \quad i = 0,\ldots,N-1$; $S[0,j] = j, \quad j = 0,\ldots,M-1$
2. Dynamic Programming
   - Paths compared using normalized cost measure

$$B[i,j] = \underset{k=1,2,3}{\arg\min} \begin{cases} \frac{D[i-1,j-1]+w_1\cdot C[i,j]}{i+j-|S[i-1,j-1]|} & \text{if } k=1 \\ \frac{D[i-1,j-2]+w_2\cdot C[i,j]}{i+j-|S[i-1,j-2]|} & \text{if } k=2 \\ \frac{D[i-2,j-1]+w_3\cdot C[i,j]}{i+j-|S[i-2,j-1]|} & \text{if } k=3 \end{cases}$$

$$D[i,j] = \begin{cases} D[i-1,j-1] + w_1\cdot C[i,j] & \text{if } B[i,j]=1 \\ D[i-1,j-2] + w_2\cdot C[i,j] & \text{if } B[i,j]=2 \\ D[i-2,j-1] + w_3\cdot C[i,j] & \text{if } B[i,j]=3 \end{cases}$$

$$S[i,j] = \begin{cases} S[i-1,j-1] & \text{if } B[i,j]=1 \\ S[i-1,j-2] & \text{if } B[i,j]=2 \\ S[i-2,j-1] & \text{if } B[i,j]=3 \end{cases}$$

3. Backtracking
   - Select best endpoint as and backtrack

$$E_{best} = \underset{(i,j)\in E_{cand}}{\arg\min} \frac{D[i,j]}{i+j-|S[i,j]|}$$

Experimental Setup

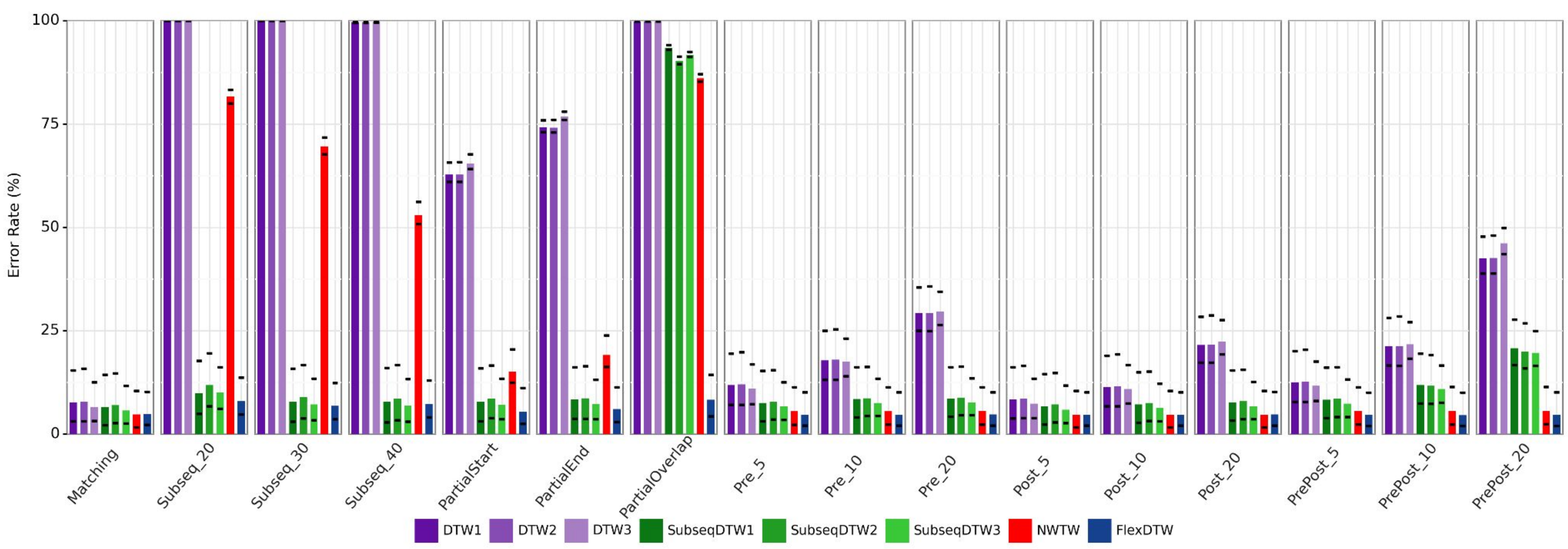
- We modified the Chopin Mazurka dataset [1] to simulate different boundary conditions. Our modifications resulted in a suite of 16 separate benchmarks, where each benchmark tests performance under a specific boundary condition.
- Boundary Conditions
  - Full Match: align full recordings of both (original dataset)
  - Subsequence: align random segment of A against full recording of B
  - Partial Start: both recordings start together but one ends early
  - Partial End: both recordings end together, but one starts late
  - Partial Overlap: recording A starts late and recording B ends early
  - Pre: silence is prepended to A and aligned against full recording of B
  - Post: silence is appended to A and aligned against full recording of B
  - Pre-Post: silence is prepended to A and appended to B

Results & Analysis

Figure shows performance of DTW (multiple settings), Subsequence DTW (multiple settings), NWTW [2], and FlexDTW on 16 different boundary conditions.

Bars shows error rate with 200ms error tolerance. Black lines show error rate with 100ms and 500ms tolerance.

FlexDTW has best or near-best performance across all 16 boundary conditions. It is the only algorithm to achieve good performance on the Partial Overlap boundary condition.



Runtime

- Average runtime (10 trials) in seconds to process a cost matrix of size NxN:

System	1k	2k	5k	10k	20k	50k
DTW	.033	0.14	0.87	3.5	13.8	87.3
SubseqDTW	0.04	0.15	0.96	3.82	15.3	96.8
NWTW	.037	0.16	0.97	3.93	15.8	101.1
FlexDTW	.038	0.16	1.05	4.21	16.9	111.1

- FlexDTW incurs a 20-25% runtime overhead compared to DTW and a 10-15% runtime overhead compared to subsequence DTW.

Memory

- FlexDTW has memory overhead for storing the additional starting point matrix $S \in \mathbb{Z}^{N\times M}$. For sequence lengths $<2^{15}$, the overhead is 2NM bytes (12% increase in total memory). For sequence lengths $>2^{15}$, the overhead is 4NM bytes (24% increase).

References & Acknowledgements

[1] C. Sapp. "Hybrid numeric/rank similarity metrics for musical performance analysis," ISMIR 2008.
[2] M. Grachten et al. "Automatic alignment of music performances with structural differences," ISMIR 2013.

This material is based upon work supported by the National Science Foundation under Grant No. 2144050.