MDPI

Article

Parameter-Free Ordered Partial Match Alignment with Hidden State Time Warping

Claire Chang [1], Thaxter Shaw [1], Arya Goutam [1], Christina Lau [2], Mengyi Shan [3] and TJ Tsai[1]*

[1] Harvey Mudd College
[2] Cornell University
[3] University of Washington
* Correspondence: ttsai@g.hmc.edu

Abstract: This paper investigates an ordered partial matching alignment problem, in which the goal is to align two sequences in the presence of potentially non-matching regions. We propose a novel parameter-free dynamic programming alignment method called hidden state time warping that allows an alignment path to switch between two different planes: a "visible" plane corresponding to matching sections and a "hidden" plane corresponding to non-matching sections. By defining two distinct planes, we can allow different types of time warping in each plane (e.g. imposing a maximum warping factor in matching regions while allowing completely unconstrained movements in non-matching regions). The resulting algorithm can determine the optimal continuous alignment path via dynamic programming, and the visible plane induces a (possibly) discontinuous alignment path containing matching regions. We show that this approach outperforms existing parameter-free methods on two different partial matching alignment problems involving speech and music.

Keywords: hidden state time warping; alignment; dynamic programming; partial matching; DTW

Citation: Chang C.; Shaw T.; Goutam A.; Lau C.; Shan M.; Tsai T. Parameter-Free Partial Match Alignment with Hidden State Time Warping. *Appl. Sci.* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Dynamic time warping (DTW) is a widely used algorithm for determining the optimal alignment between two sequences. Standard DTW assumes that the two sequences start and end at the same location, and that the time warping is defined by a predetermined set of allowable transitions. Variations of DTW have been proposed to estimate alignments under various assumptions and conditions. For example, subsequence DTW determines the optimal alignment between a short query sequence and any subsequence within a longer reference sequence. In this paper, we propose a time warping algorithm for handling ordered partial alignments, where the true alignment between two sequences may contain discontinuities due to non-matching regions but the matching regions are assumed to be ordered. Figure 1 provides a graphical illustration of the difference between the full matching, subsequence matching, and ordered partial matching scenarios.

For the sake of completeness, we provide an overview of standard DTW before describing variations and extensions of it in the literature. Estimating the alignment between two sequences $x_1, x_2, \ldots, x_N$ and $y_1, y_2, \ldots, y_M$ with DTW consists of three main steps. The first step is to compute a pairwise cost matrix $C \in \mathbb{R}^{N \times M}$, where $C[i, j]$ indicates the dissimilarity between x_i and y_j. The second step is to compute a cumulative cost matrix $D \in \mathbb{R}^{N \times M}$, where $D[i, j]$ indicates the lowest cost cumulative path score starting at $C[0, 0]$ and ending at $C[i, j]$. The elements in D can be computed using dynamic programming based on a set of allowable transitions in the cost matrix, and a separate backtrace matrix $B \in \mathbb{Z}^{N \times M}$ keeps track of the last transition in the lowest cost path ending at each location. The third step is to follow the backtrace pointers in B to determine the lowest cost path from $C[0, 0]$ to $C[N-1, M-1]$.

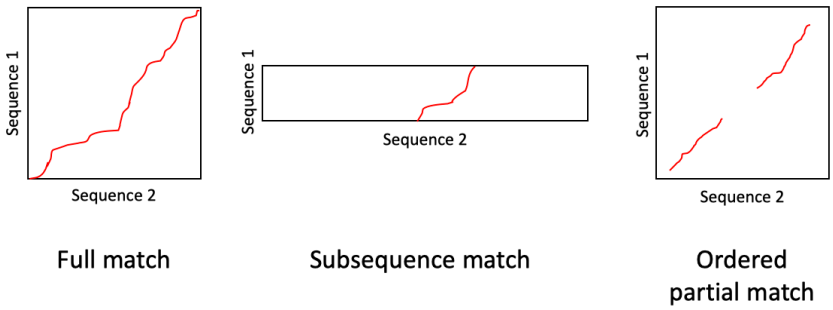

Figure 1. Different types of alignment problems. HSTW provides a parameter-free solution to the ordered partial match alignment problem.

Many works have proposed variations or extensions of DTW.[1] These works generally fall into one of two categories. The first category of works focuses on mitigating the quadratic runtime or memory costs of DTW, which make it prohibitive for processing long sequences. Some works approach this by proposing approximations of DTW, such as by imposing fixed boundaries on the alignment path (e.g. Sakoe-Chiba band [1], Itakura parallelogram [2]), estimating the alignment at a coarse resolution and iteratively refining the alignment path [3][4], estimating the alignment path with a limited amount of memory [5], or using a parallelizable time warping algorithm [6]. Other works approach this by computing exact DTW in a more efficient way, such as by using lower bounds [7][8], early abandoning [9][10], utilizing multiple cores [11][12] or specialized hardware [13][14], or performing the dynamic programming along diagonals rather than rows/columns to allow for parallelization [15]. The second category of works focus on extending the behavior of DTW in some way. Some examples in the audio processing literature include performing time warping in an online fashion [16][17], handling repeats and jumps in music [18][19][20], handling subsequence alignments [21][22], handling pitch drift in a capella performances [23], and utilizing multiple performances to improve alignment accuracy [24].

This paper proposes an alignment algorithm called hidden state time warping (HSTW) that can handle ordered partial matching. In particular, HSTW allows for aligning two sequences whose alignment may contain discontinuities due to non-matching regions, as shown in Figure 1. Rather than computing a cumulative cost matrix $D \in \mathbb{R}^{N \times M}$, we instead compute a cumulative cost tensor $D \in \mathbb{R}^{2 \times N \times M}$ containing two different planes: a "visible" plane corresponding to matching sections and a "hidden" plane corresponding to non-matching sections. The alignment path can transition between the visible and hidden planes. By defining two distinct planes, we can use different dynamic programming rules to treat matching and non-matching sections differently (e.g. imposing a maximum time warping factor in matching sections while allowing unconstrained movements in non-matching sections). Furthermore, we can find the optimal (continuous) alignment path through the entire double-plane using a dynamic programming framework, and then induce a discontinuous optimal matching alignment path by considering only the path elements in the visible plane. In the following sections, we describe this algorithm in more detail and provide experimental evidence of its effectiveness on two different tasks:

[1] It is worth mentioning how DTW fits into the broader literature of alignment algorithms. At the highest level of abstraction, alignment algorithms can be categorized into two groups: those with trainable parameters (e.g. HMMs) and those without trainable parameters (parameter-free algorithms like DTW which only have hyperparameters). Both groups use dynamic programming to estimate an alignment, but they differ in the contexts in which they are used. HMMs can only be used when training data is available and are the preferred method when aligning unstructured data to a sequence of symbolic symbols (e.g. aligning music audio to a symbolic score or aligning speech audio to a sequence of words). In contrast, DTW can be used without any training data and is usually the preferred method when aligning two sequences of unstructured data (e.g. audio-audio alignment). In this paper, we focus exclusively on parameter-free algorithms. The main advantage of a parameter-free method is its simplicity and "portability": it can be applied to any type of data without training, as we demonstrate on two different tasks in this paper.

one involving the alignment of speech data in the presence of tampering operations like insertion, deletion, and replacement, and another involving the synchronization of music performances with different content (e.g. silence or applause) before, after, and in between movements of a musical piece. [2]

2. Materials and Methods

In this section, we describe how to align two partially matching audio recordings using our proposed hidden state time warping (HSTW) algorithm. There are three steps: extracting the features, estimating a global alignment, and determining which regions are matching. These three steps will be discussed in the next three subsections (Sections 2.1 - 2.3), followed by a discussion of initialization (Section 2.4) and setting hyperparameters (Section 2.5).

2.1. Feature Extraction

The first step is to extract features from both audio recordings. HSTW can work with any feature representation, so for simplicity we use standard feature representations. For speech audio, we compute 13-dimensional MFCCs with delta and delta-delta features (total 39 dimensions) using 25 ms analysis frames and 10 ms hop size. For music audio, we compute 12-dimensional chroma features based on a constant Q transform with 23 ms hop size. At the end of this first step, we have two sequences of features $x_1, \ldots, x_{L_1}$ and $y_1, \ldots, y_{L_2}$.

2.2. Alignment

The second step is to estimate a global continuous alignment path between the two sequences of features. This process consists of three sub-steps, which are described in the next three paragraphs.

First, we compute a pairwise cost matrix $C \in \mathbb{R}^{L_1 \times L_2}$ between the two feature sequences. For speech audio we use Euclidean distance as a dissimilarity metric, and for music audio we use cosine distance. So, $C[i, j]$ indicates the dissimilarity between the feature x_i in the first sequence and the feature y_j in the second sequence.

Second, we compute a cumulative cost tensor $D \in \mathbb{R}^{2 \times L_1 \times L_2}$ (and corresponding backtrace tensor $B \in \mathbb{N}^{2 \times L_1 \times L_2}$) using the following dynamic programming rules:

$$D_h[i,j] = \begin{cases} \frac{\gamma+\alpha}{2} & i=0, j=0 \\ \min \begin{cases} D_h[i,j-1] + \gamma \\ D_h[i-1,j] + \alpha \\ D_v[i-1,j-1] + \gamma + \alpha \end{cases} & \text{otherwise} \end{cases}$$

$$D_v[i,j] = \begin{cases} C[0,0] & i=0, j=0 \\ \min \begin{cases} D_v[i-1,j-1] + 2 \cdot C[i,j] \\ D_v[i-2,j-1] + 3 \cdot C[i,j] \\ D_v[i-1,j-2] + 3 \cdot C[i,j] \\ D_h[i,j] + \beta \end{cases} & \text{otherwise} \end{cases}$$

where D_h and D_v correspond to the hidden and visible planes, respectively. In the multi-case expressions for both $D_h[i, j]$ and $D_v[i, j]$, the first case handles the initialization of the cumulative cost tensor, and the second case specifies the dynamic programming rules. There are three types of transitions that end in the hidden plane: (i) a horizontal transition to the right in the hidden plane with a penalty of γ, (ii) a vertical transition upwards in the hidden plane with a penalty of α, and (iii) a diagonal transition from position $(i-1, j-1)$ in the visible plane to position (i, j) in the hidden plane with penalty $\gamma + \alpha$. Note that

[2] Code for reproducing the results can be found at https://github.com/HMC-MIR/HSTW.

these three cases all incur fixed penalties that are independent of the pairwise cost $C[i,j]$. There are four types of transitions that end in the visible plane: three regular transitions $\{(1,1),(1,2),(2,1)\}$ in the visible plane with corresponding transition weights $\{2,3,3\}$, as well as a transition from position (i,j) in the hidden plane to position (i,j) in the visible plane with a penalty of β. As we compute each element of $D_h[i,j]$ and $D_v[i,j]$, we also update the corresponding backtrace tensor entries $B_h[i,j]$ and $B_v[i,j]$ to record the optimal transition type at each position.

Third, we determine the optimal global continuous alignment path through D using backtracking. We start with the element in the last row and column of D_v or D_h (whichever has a lower cumulative path score), and then follow the back pointers in B to determine each step of the optimal path. We stop backtracking once we have reached either $D_v[0,0]$ or $D_h[0,0]$.

2.3. Classification

The third step is to classify each element in the optimal global continuous alignment path as matching or not matching. The classification for a path element can then be imputed to the corresponding audio frames in the two recordings. For example, if a path element (i,j) is classified as a non-match, we can infer that the i^{th} frame of the first recording does not match the other recording. With the HSTW algorithm, the classification can be done in two ways: (1) classify path elements in the visible plane as matches and path elements in the hidden plane non-matches, or (2) classify each path element (i,j) by applying a hard threshold to the corresponding pairwise cost $C[i,j]$. We report results with both classification methods.

2.4. Initialization

The equations shown above are not the HSTW algorithm – they are just one *example* of HSTW. In the same way that DTW can be modified by choosing a different set of allowable transitions and weights or modified to handle subsequence matches, so too can HSTW be flexibly adapted to a range of scenarios.

One way to modify HSTW behavior is to choose different initialization conditions for the cumulative cost tensor D. For example, the equations can be modified to handle subsequence alignment by introducing two changes: (a) initializing $D_v[0,j] = C[0,j]$, $j = 0,1,\ldots,L_2-1$ so that the alignment path can begin anywhere in the second (longer) sequence without penalty and (b) beginning the backtracking from position (L_1-1,j^*) where $j^* = \text{argmin}_j D_v[L_1-1,j]$ to allow the alignment path to end anywhere without penalty. One can treat D_h similarly to allow subsequence paths to start or end in the hidden plane.

Another way to modify HSTW behavior is to choose different allowable transitions and transition weights. The equations shown above allow for $(1,1)$, $(1,2)$, and $(2,1)$ transitions within the visible plane with corresponding transition weights 2, 3, and 3. These allowable transitions and transition weights could be chosen differently to encourage or allow certain types of alignment paths.

2.5. Hyperparameters

In addition to the hyperparameters that DTW has, HSTW has three additional hyperparameters: γ, α, and β. These three hyperparameters have clear interpretations: γ is a skip penalty for a horizontal transition, α is a skip penalty for a vertical transition, and β is a plane transition penalty. These hyperparameters can be set to achieve different types of behavior. We describe our method of setting these hyperparameters for our two tasks of interest.

We set the hyperparameters on our (subsequence alignment) speech verification task to perform well in the presence of common tampering operations like insertions, deletions, and replacements. For the deletion penalty γ, we found that the optimal setting is a small, positive number. We want the cost to be small in order to allow skipping in the reference

Piece	Files	mean	std	min	max
Opus 17, No 4	64	259.7	32.5	194.4	409.6
Opus 24, No 2	64	137.5	13.9	109.6	180.0
Opus 30, No 2	34	85.0	9.2	68.0	99.0
Opus 63, No 3	88	129.0	13.4	96.2	162.9
Opus 68, No 3	51	101.1	19.4	71.8	164.8

Table 1. Overview of the Chopin Mazurka data used in the music alignment task. All durations are in seconds. The top two pieces are used for validation (i.e. setting hyperparameters), and the bottom three pieces are used for testing. Tampered versions of the performances are generated to simulate non-matching regions at the beginning, middle, and end of the recording.

dimension when there is a deletion edit in the query. If we set γ to zero, however, it allows for arbitrarily long skips, which has undesirable behavior. We use $\gamma = 3$ in all of our reported results, and we find that system performance is relatively insensitive for values within the range $0.5 < \gamma < 10$. For the insertion penalty α, we found that the optimal value strongly depends on the values in the cost matrix. If α is too large, the algorithm cannot follow vertical alignment paths corresponding to insertions. If α is too small, the optimal subsequence path consists of simply skipping across the entire cost matrix. Ideally, we want the penalty to be greater than the cost between two matching elements but smaller than the cost between non-matching elements. Accordingly, we set α in a data-dependent way: we calculate the minimum cost in each row of the cost matrix C (i.e. the minimum cost for each query frame), and then set α to $c_\alpha = 2.4$ times the median of these row minima. We find system performance to be insensitive across the range $2 < c_\alpha < 3$. For the hidden-to-visible transition penalty β, we find that $\beta = c_\beta * (\gamma + \alpha)$ with $c_\beta = 33$ works well, and that system performance is insensitive for values within the range $30 < c_\beta < 40$. This penalty prevents the alignment path from coming up to the visible plane unless it can "consume" enough low-cost elements.

We set the hyperparameters on our music alignment task in the same manner but with one difference: we always set $\gamma = \alpha$ since the alignment task is symmetric. Based on our training data, we used $c_\gamma = 10$ and $c_\beta = 150$.

3. Results

In this section we describe a set of experiments with two different tasks involving both music and speech data. We first describe the experimental setup of both tasks, and then present our empirical results.

3.1. Experimental Setup - Music Synchronization

The first task is synchronization of different music performances of the same piece in the presence of non-matching regions. We use the Mazurka dataset [25], which consists of multiple performances of five different Chopin Mazurkas. For each performance, the dataset contains manual annotations of beat-level timestamps. We set aside two Mazurkas for training and the other three Mazurkas for testing, resulting in a total of 2514 training alignment pairs and 7069 test alignment pairs. This dataset has been used in a number of previous studies on music alignment [26][6][27][25]. Table 1 shows an overview of the Mazurka dataset.

There are a number of factors that might cause two classical music recordings of the same piece to have non-matching regions. For example, a CD audio recording might start and end at the beginning and end of the actual performance, while a Youtube recording might contain silence or applause at the beginning and/or end of the video. Non-matching regions in the middle of a recording might be due to structural differences like repeats, silence between movements of a piece, or a different cadenza.

We generate tampered versions of the Mazurka performances in order to study the effect of non-matching regions. The beginning, middle, and end of each original Mazurka performance is tampered in the following manner. First, a filler audio recording of a *different* Mazurka is randomly selected from the same train-test split. This filler recording is used to replace original content with non-matching content. Second, the first L_{start} seconds of the original recording is replaced with a random interval selected from the filler recording, where L_{start} is selected uniformly in the interval $[0, 10]$ seconds. Third, the last L_{end} seconds of the original recording is likewise replaced with a random interval in the filler recording, where L_{end} is selected uniformly from $[0, 10]$ seconds. Finally, a random L second interval in the middle of the original recording is replaced with content from the filler recording. We make sure the middle tampered region is at least 10 seconds away from the tampered regions at the beginning and end, so as to avoid creating very short matching fragments. Each tampered recording is aligned against all other (untampered) original recordings of the same Mazurka.

We evaluate system performance based on how well it classifies each audio frame in the query (i.e. the tampered recording) as matching or non-matching. Note that determining whether or not an audio frame is matching or non-matching requires solving the alignment problem accurately.[3] We characterize performance using a standard receiver operating characteristic (ROC) curve, which shows the tradeoff between false positives and false negatives. When evaluating classification performance, we exclude frames that lie within a $\pm\Delta = 100$ ms scoring collar around tampering locations.

3.2. Experimental Setup - Speech Alignment & Verification

The second task is verifying the legitimacy of a short speech recording (e.g. a viral video clip of a presidential speech on social media) by comparing it to a trusted reference recording (e.g. an official recording of the entire speech from a major news outlet). The goal is to be able to perform verification in the presence of common tampering operations like insertions, deletions, and replacements. Note that while there is a rich literature in audio tampering detection by detecting internal inconsistencies and tampering artifacts (see [28] for a survey), our task instead focuses exclusively on *positively* verifying the legitimacy of an audio recording by aligning and comparing it to a trusted external reference recording. This problem has become particularly relevant in recent years, as digital audio editing software and deep fake technology has become readily accessible [29].

To simulate a realistic scenario involving a high-profile political figure, we use recordings of former U.S. President Donald Trump. We collected 50 speech recordings of Trump from 2017 to 2018, all taken from the official White House Youtube playlist when he was in office. Each of these recordings is single-speaker and is 1-3 minutes long.

We generate audio queries from the raw data in the following manner. We randomly select ten 10-second segments from each recording, and then generate four different versions of each segment by applying various tampering operations. The first version has no tampering: it is simply an unedited version at a lower bitrate quality (160 kbps) than the reference (320 kbps). The second version has an insertion: we randomly select an L second fragment from a different audio recording (within the dataset) and insert the fragment into the 10-second segment at a random location. The third version has a deletion: we randomly select an L second interval from the segment and delete it. The fourth version has a replacement: we randomly select an L second fragment from a different audio recording and use it to replace a randomly selected L second fragment in the 10-second segment. For a given value of L, this process will generate a total of $4 \times 50 \times 10 = 2000$ queries. We divide the queries by speech recording and use 50% for training and 50% for testing. Each query is aligned against the corresponding high-resolution reference audio recording from which it was taken.

[3] We found that all systems had roughly the same alignment accuracy, so we only focus on reporting classification performance.

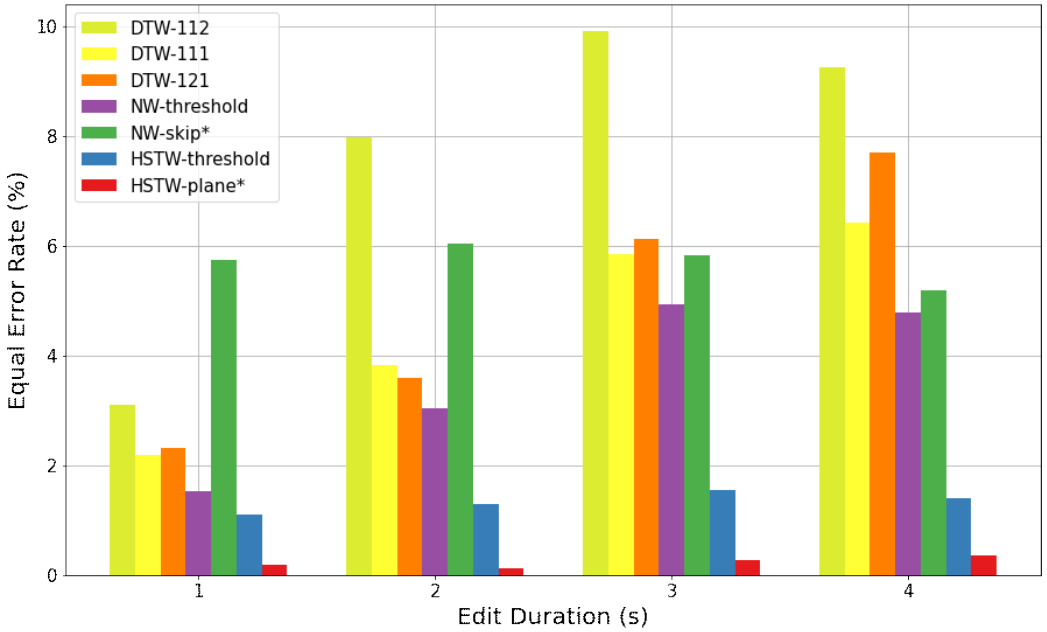

Figure 2. Results on the speech alignment & verification task. Bars indicate the equal error rate for classifying audio frames as matching vs. non-matching. Systems with * are not threshold-based and therefore do not sweep out a full ROC curve. For these systems, the bar indicates the mean of the false positive and false negative rate.

We evaluate our speech verification system based on how well it classifies each audio frame in the query as matching or non-matching. Again, we note that solving the classification task requires solving the alignment task accurately, and is therefore a better indicator of system-level performance. We evaluate this binary classification task using a standard ROC curve and corresponding equal error rate (EER). We exclude frames that lie within a $\pm\Delta = 100$ ms scoring collar around tampering locations.

3.3. Speech Alignment & Verification Results

For the speech task, we compare the performance of seven different parameter-free alignment systems. The first system ("DTW-111") is subsequence DTW with (query, reference) transitions $\{(1,1),(0,1),(1,0)\}$ and corresponding transition weights $\{1,1,1\}$. For each element (i,j) on the optimal alignment path, the pairwise cost $C[i,j]$ is imputed as a score to the i^{th} frame in the query (i.e. the potentially tampered recording), and each query frame's score is compared to a threshold to determine if the frame is a match or a non-match. This threshold can be varied to sweep out an ROC curve. The second ("DTW-121") and third ("DTW-112") systems are the same as the first system but with transition weights $\{1,2,1\}$ and $\{1,1,2\}$, respectively. The fourth system ("NW-threshold") is Needleman-Wunsch alignment [30], which allows for regular $(1,1)$ transitions as well as $(0,1)$ and $(1,0)$ transitions with a fixed insertion/deletion penalty γ that does not depend on the pairwise cost matrix C. Once the optimal alignment path has been estimated, scores are imputed to the query frames and thresholded, as before. The fifth system ("NW-skip") is identical to the fourth system but simply classifies insertions and deletions as non-matches and $(1,1)$ transitions as matches. The sixth system ("HSTW-threshold") uses our proposed algorithm to estimate an optimal subsequence alignment path, imputes scores to each query frame based on the pairwise cost matrix, and then applies a threshold. The seventh system ("HSTW-plane") estimates the alignment with our proposed algorithm and simply classifies path elements in the visible plane as matches and path elements in the hidden plane as non-matches. For each system above, hyperparameters are tuned on the training set.

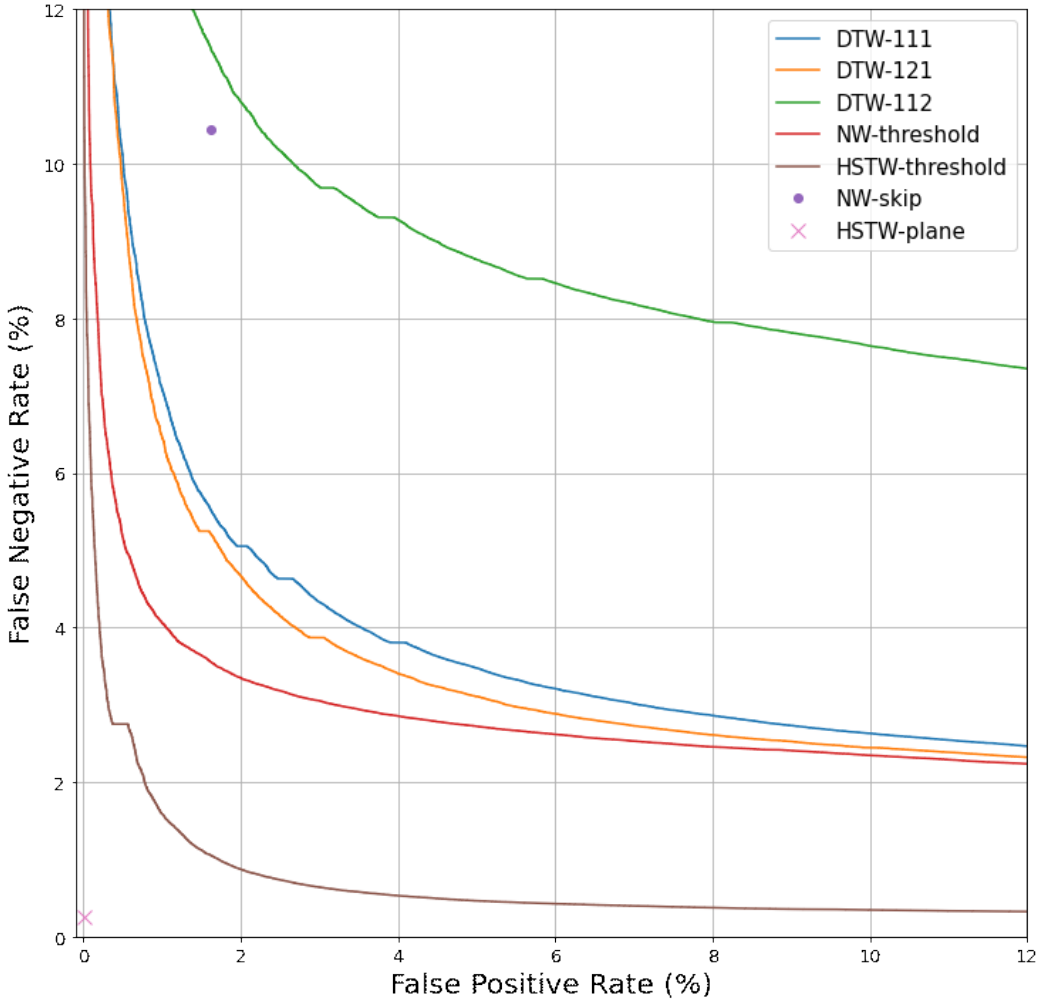

Figure 3. ROC curves for all systems on the speech alignment & verification task with $L = 2$s tampering duration. NW-skip and HSTW-plane are not threshold-based systems, so their performance is characterized by a single point.

Figure 2 shows the frame classification results on the speech verification task for $L = 1, 2, 3, 4$s tampering durations. For all threshold-based systems (i.e. all systems except NW-skip and HSTW-plane), the bars indicate the EER of classification performance. Because NW-skip and HSTW-plane are not threshold-based systems, their performance is defined by a single point in the ROC plane, as shown in Figure 3 for $L = 2$s tampering durations. For these two systems, an approximation of EER is estimated by averaging the false positive and false negative rate.

There are three things to notice about Figures 2 and 3. First, HSTW-plane has the best classification performance by a wide margin. For example, for $L = 2$s tampering operations, HSTW-plane achieves 0.01% false positive rate and 0.25% false negative rate, while the best non-HSTW system achieves 3.04% equal error rate. Second, HSTW-plane performs much better than HSTW-threshold, even though both systems have identical alignment paths. This indicates that there is relevant information in the visible/hidden dimension that is not captured by the 2D alignment between the two sequences. Third, the performance of non-HSTW systems varies drastically based on the tampering duration, but is much more stable for the HSTW-based systems.

3.4. Music Synchronization Results

For the music task, we compare the performance of six different systems. The first two systems ("DTW-233", "DTW-111") are standard DTW with transitions $\{(1,1),(1,2),(2,1)\}$ and corresponding transition weights $\{2,3,3\}$ or $\{1,1,1\}$. Scores are imputed to each frame in the query (i.e. the tampered recording) and thresholded, as before. The third system ("NWTW-threshold") was proposed in [20] as a way to handle both time warping as well as insertions and deletions. It allows for $\{(1,1),(1,2),(2,1)\}$ transitions to handle time warping, as well as $(0,1)$ and $(1,0)$ skip transitions with a fixed penalty γ. Scores are imputed to the query frames and thresholded as above. The fourth system ("NWTW-skip") is identical to the third system but simply classifies insertion and deletion transitions as non-matches and all other regular transitions as matches. The fifth system ("HSTW-threshold") uses our proposed algorithm to estimate an optimal global alignment path, and then imputes scores to each query frame and thresholds as before. The sixth system ("HSTW-plane") estimates the alignment with our proposed algorithm and simply classifies path elements in the visible plane as matches and path elements in the hidden plane as non-matches. For each system above, hyperparameters are tuned on the training set.

Figure 4 shows the results on the music synchronization task with $L = 5, 10, 20, 30$s tampering durations. We see that HSTW-plane again has the best performance, but that the music task is much more challenging than the speech task based on the equal error rates. This is because the speech task involves comparing two exactly matching recordings (with possible tampering operations), whereas the music task involves comparing two different performances of a piece (with additional tampering operations). Again, we see that HSTW-plane has better performance than HSTW-threshold despite the fact that both share the same predicted alignment path. This suggests that the hidden and visible planes are capturing important and relevant information that the 2D alignment path is not capturing.

Figure 5 shows two example HSTW alignments from the music task that illustrate when HSTW performs well (top) and when HSTW performs poorly (bottom). The green and red regions indicate ground truth matching and non-matching regions in time (i.e. the red regions correspond to tampered operations), the blue line shows which plane (hidden or visible) each element of the optimal HSTW path is in, and the black line shows the corresponding costs in the pairwise cost matrix. We can see that HSTW correctly identifies the tampered regions: it transitions from the visible plane to the hidden plane when there is tampering. In the bottom example, we can see that the matching and non-matching regions are difficult to differentiate based on pairwise cost values, which leads to higher error rates compared to the speech task. Looking at the pairwise costs, the HSTW predictions are reasonable though incorrect.

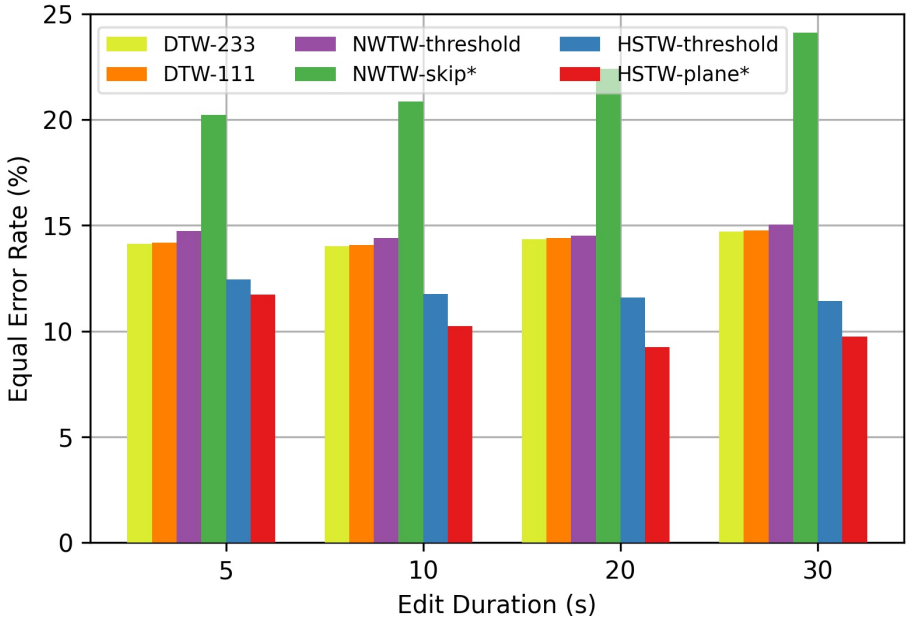

Figure 4. Results on the music synchronization task with non-matching regions. Bars indicate the equal error rate for classifying audio frames as matching vs. non-matching. For systems that are not threshold-based (indicated with *), bars indicate the mean of the false positive and false negative rate.

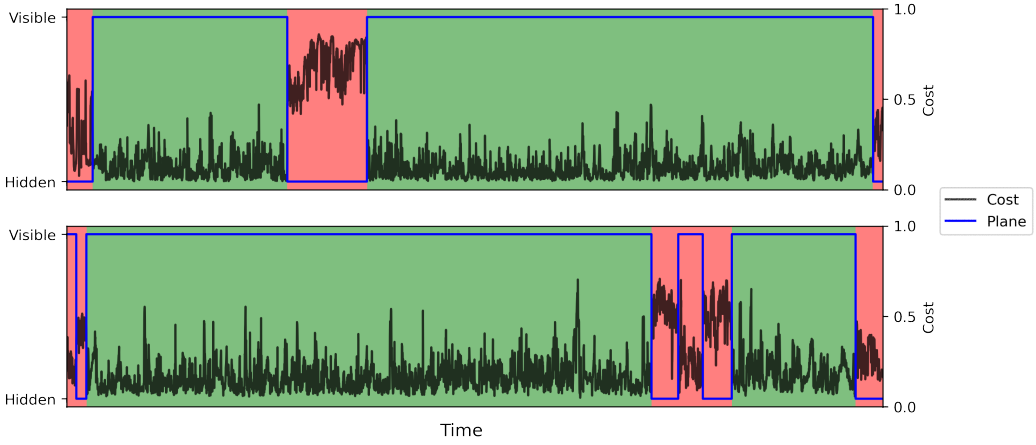

Figure 5. Two example HSTW alignments on the music task. Green and red regions indicate ground truth matching and non-matching regions, respectively. The blue line shows the hidden/visible plane of the optimal path, and the black line shows corresponding costs in the pairwise cost matrix.

4. Discussion

In this section, we discuss the benefits of HSTW compared to previous work, its limitations, and areas of future work.

The benefits of HSTW can be viewed from two different perspectives. As mentioned earlier, alignment algorithms can be categorized into two groups: trainable alignment methods like HMMs and parameter-free alignment methods like DTW. Compared to trainable alignment methods, the main benefit of HSTW is its simplicity and portability: it can be easily adapted to new domains without requiring a cumbersome training process, as we demonstrated with two different tasks in this article. Compared to previous parameter-free alignment methods, the main benefit of HSTW is in its improved handling of non-matching regions (which results in an ordered partial match alignment problem). The most relevant previous parameter-free method is NWTW [20], which allows for horizontal/vertical skips as well as time warping transitions at each position in the cost matrix. While NWTW should in theory be able to handle non-matching regions as a consecutive sequence of horizontal and vertical skips, we see from Figures 2 and 4 that it performs substantially worse than HSTW in handling non-matching regions. Since NWTW and HSTW both allow for the exact same types of transitions, the difference in performance comes from separating the single state (in NWTW) into two distinct states with different time warping characteristics.

There are two limitations or drawbacks to using HSTW. The first drawback is the additional computation and memory requirements compared to standard DTW. Because of the additional hidden plane in the cumulative cost matrix and backtrace matrix, HSTW requires twice as much computation and memory as DTW with the same size cost matrix. So, while HSTW is in theory a generalization of DTW that could be used as a more flexible alternative, in practice its use is only recommended for handling ordered partial match alignment. The second drawback is the need to set 2-3 additional hyperparameters[4] beyond those of standard DTW. This can be done with a very small amount of data, however, so this would be considered a relatively minor drawback.

For future work, we plan to generalize HSTW to more than two states. While HSTW explores the use of a hidden and visible state, the concept can readily be extended to multiple states, each having its own time warping characteristics. This would allow one to encode structure a priori into the alignment algorithm while remaining a parameter-free algorithm.

Author Contributions: Conceptualization, T.T.; methodology, M.S. and T.T.; software, C.C., T.S., A.G., C.L., and M.S.; validation, C.C., T.S., A.G., C.L., and M.S.; writing—original draft preparation, T.T.; writing—review and editing, C.C., T.S., A.G., C.L., and M.S.; supervision, T.T.; funding acquisition, T.T. All authors have read and agreed to the published version of the manuscript.

Funding: This material is based upon work supported by the National Science Foundation under Grant No. 1948531.

Data Availability Statement: Non-copyrighted data for our experiments can be found at https://github.com/HMC-MIR/HSTW.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

[4] Recall that there are three hyperparameters: the skip penalty in the horizontal direction, the skip penalty in the vertical direction, and the plane transition penalty. If the alignment problem is symmetric, however, only two effective hyperparameters need to be set.

DTW	Dynamic time warping
HSTW	Hidden state time warping
HMMs	Hidden markov models
MFCCs	Mel frequency cepstral coefficients
ROC	Receiver operating characteristic
kbps	kilobits per second
EER	Equal error rate
NWTW	Needleman-Wunsch time warping

References

1. Sakoe, H.; Chiba, S. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **1978**, *26*, 43–49.
2. Itakura, F. Minimum Prediction Residual Principle Applied to Speech Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **1975**, *23*, 67–72.
3. Salvador, S.; Chan, P. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. Proc. of the KDD Workshop on Mining Temporal and Sequential Data, 2004.
4. Müller, M.; Mattes, H.; Kurth, F. An Efficient Multiscale Approach to Audio Synchronization. Proc. of the International Conference on Music Information Retrieval (ISMIR), 2006, pp. 192–197.
5. Prätzlich, T.; Driedger, J.; Müller, M. Memory-Restricted Multiscale Dynamic Time Warping. Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2016, pp. 569–573.
6. Tsai, T. Segmental DTW: A Parallelizable Alternative to Dynamic Time Warping. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 106–110.
7. Zhang, Y.; Glass, J. An Inner-Product Lower-Bound Estimate for Dynamic Time Warping. Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2011, pp. 5660–5663.
8. Keogh, E.; Wei, L.; Xi, X.; Vlachos, M.; Lee, S.H.; Protopapas, P. Supporting Exact Indexing of Arbitrarily Rotated Shapes and Periodic Time Series under Euclidean and Warping Distance Measures. *VLDB Journal* **2009**, *18*, 611–630.
9. Rakthanmanon, T.; Campana, B.; Mueen, A.; Batista, G.; Westover, B.; Zhu, Q.; Zakaria, J.; Keogh, E. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 262–270.
10. Li, J.; Wang, Y. EA DTW: Early Abandon to Accelerate Exactly Warping Matching of Time Series. International Conference on Intelligent Systems and Knowledge Engineering, 2007.
11. Srikanthan, S.; Kumar, A.; Gupta, R. Implementing the Dynamic Time Warping Algorithm in Multithreaded Environments for Real Time and Unsupervised Pattern Discovery. International Conference on Computer and Communication Technology, 2011, pp. 394–398.
12. Shabib, A.; Narang, A.; Niddodi, C.P.; Das, M.; Pradeep, R.; Shenoy, V.; Auradkar, P.; Vignesh, T.; Sitaram, D. Parallelization of Searching and Mining Time Series Data using Dynamic Time Warping. IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2015, pp. 343–348.
13. Sart, D.; Mueen, A.; Najjar, W.; Keogh, E.; Niennattrakul, V. Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs. IEEE International Conference on Data Mining, 2010, pp. 1001–1006.
14. Wang, Z.; Huang, S.; Wang, L.; Li, H.; Wang, Y.; Yang, H. Accelerating Subsequence Similarity Search Based on Dynamic Time Warping Distance with FPGA. Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2013, pp. 53–62.
15. Tralie, C.J.; Dempsey, E. Exact, Parallelizable Dynamic Time Warping Alignment with Linear Memory. Proc. of the International Conference for Music Information Retrieval (ISMIR), 2020, pp. 462–469.
16. Macrae, R.; Dixon, S. Accurate Real-time Windowed Time Warping. Proc. of the International Conference on Music Information Retrieval (ISMIR), 2010, pp. 423–428.
17. Dixon, S. Live Tracking of Musical Performances Using On-line Time Warping. Proc. of the International Conference on Digital Audio Effects, 2005, pp. 92–97.
18. Fremerey, C.; Müller, M.; Clausen, M. Handling Repeats and Jumps in Score-Performance Synchronization. Proc. of the International Conference on Music Information Retrieval (ISMIR), 2010, pp. 243–248.
19. Shan, M.; Tsai, T. Improved Handling of Repeats and Jumps in Audio-Sheet Image Synchronization. Proc. of the International Society for Music Information Retrieval (ISMIR), 2020, pp. 62–69.
20. Grachten, M.; Gasser, M.; Arzt, A.; Widmer, G. Automatic Alignment of Music Performances with Structural Differences. Proc. of the International Society for Music Information Retrieval Conference (ISMIR), 2013.
21. Müller, M.; Appelt, D. Path-Constrained Partial Music Synchronization. Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2008, pp. 65–68.
22. Müller, M.; Ewert, S. Joint Structure Analysis with Applications to Music Annotation and Synchronization. Proc. of the International Society for Music Information Retrieval Conference (ISMIR), 2008, pp. 389–394.

23. Waloschek, S.; Hadjakos, A. Driftin' Down the Scale: Dynamic Time Warping in the Presence of Pitch Drift and Transpositions. Proc. of the International Conference on Music Information Retrieval (ISMIR), 2018, pp. 630–636.
24. Wang, S.; Ewert, S.; Dixon, S. Robust and Efficient Joint Alignment of Multiple Musical Performances. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **2016**, *24*, 2132–2145.
25. Sapp, C. Hybrid Numeric/Rank Similarity Metrics for Musical Performance Analysis. Proc. of the International Conference for Music Information Retrieval (ISMIR), 2008, pp. 501–506.
26. Schreiber, H.; Zalkow, F.; Müller, M. Modeling and Estimating Local Tempo: a Case Study on Chopin's Mazurkas. Proc. of the International Society for Music Information Retrieval Conference (ISMIR), 2020, pp. 773–779.
27. Grosche, P.; Müller, M.; Sapp, C.S. What Makes Beat Tracking Difficult? A Case Study on Chopin Mazurkas. Proc. of the International Conference for Music Information Retrieval (ISMIR), 2010, pp. 649–654.
28. Zakariah, M.; Khan, M.K.; Malik, H. Digital Multimedia Audio Forensics: Past, Present and Future. *Multimedia Tools and Applications* **2018**, *77*, 1009–1040.
29. Chesney, B.; Citron, D. Deep Fakes: A Looming Challenge for Privacy, Democracy, and National Security. *Calif. L. Rev.* **2019**, *107*, 1753.
30. Needleman, S.B.; Wunsch, C.D. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology* **1970**, *48*, 443–453.