k-Best Hidden Markov Model Decoding for Unit Selection in Concatenative Sound Synthesis

Cárthach Ó Nuanáin, Sergi Jordà and Perfecto Herrera

Music Technology Group
Universitat Pompeu Fabra
Barcelona
carthach.onuanain@upf.edu

Abstract. Concatenative synthesis is a sample-based approach to sound creation used frequently in speech synthesis and, increasingly, in musical contexts. Unit selection, a key component, is the process by which sounds are chosen from the corpus of samples. Hidden Markov Models are often chosen for this task, but one common criticism is its singular path output which is considered too restrictive when variations are desired. In this paper, we propose considering the problem in terms of *k*-Best path solving for generating alternative lists of candidate solutions and summarise our implementations along with some examples.

Keywords: Hidden Markov Models, Concatenative Synthesis, Artificial Intelligence, Musical Signal Processing

1 Introduction

Concatenative synthesis is a technique that generates new sounds by juxtaposing existing sounds from a large collection or "corpus". It was first applied in the area of speech synthesis [8], but has since been extended to musical and sound design tasks [23]. It is closely related to granular synthesis, but differs in the order of scale of the length of the sounds that are used. Granular synthesis typically operates on the microscale with “grains” of lengths 20-200 ms [17], whereas concatenative synthesis makes use of samples of unit lengths more musically related, such as a note or a phrase. An inherently Music Information Retrieval (MIR) geared approach, feature extraction of acoustic and musical descriptors (such as spectral, energy and timbral features) are essential for analysing and sorting existing sounds then concatenating them to create new ones according to some predefined strategy.

The unit selection procedure of concatenative synthesis is the process by which the existing sounds are selected from the corpus and is typically based on feature analysis to match the characteristics of some target sound or specification. Many algorithms have been proposed for tackling unit selection, but one of the most well-known involves the application of Hidden Markov Models (HMMs) and in particular Viterbi decoding [20] of state sequences to produce a stochastically optimal output sequence of concatenated sounds. As we will discuss in the forthcoming section, Viterbi decoding of HMMs has its limitations [20], chief of which is the fact that it outputs only the highest probable state sequence.

In many objective problem applications (such as route finding in a network) this is sufficient and objectively defined, but in sound synthesis and particularly highly subjective musical and compositional tasks we would rather produce subsequent probabilistic sequences in order to explore and evaluate alternative possibilities. To this end, we present in this article methods for reformulating and extending the well-known Viterbi decoding algorithm to handle generating the *k*-best candidate sequences [21], and describe how this can then be used in a practical context of musical concatenative sound synthesis.

2 Markov Models

Markov chains are probabilistic state machines that satisfy the Markov property of *memorylessness*. They consist of a set of discrete states with each state having an associated probability weighting of moving to every other state in the system. At any point in discrete time, the probability of a future event is solely determined by the current state of the system without knowledge of past events.

A hidden markov model $\lambda = (A, B, \pi)$ extends the concept of a Markov chain by considering the transition states as *hidden* [16]. The hidden states have a transition matrix *A* as before, but each hidden state also emits an *observable* symbol from a set of symbols that have a probability distribution encapsulated in an emission matrix *B*. Finally, to initiate the HMM there also exists the initial probability distribution π, which determines the probability of which state to commence.

2.3 The Viterbi Algorithm

The Viterbi algorithm solves the decoding problem in HMMs [16], namely, for a given observation sequence $O = (O_1 O_2 O_3 \ldots O_T)$ we wish to determine the highest probable hidden state sequence $S = (S_1 S_2 S_3 \ldots S_T)$ that would produce *O*.

A brute force solution applied to a set of *T* observations over *N* states would involve computing all the Cartesian products of the possibilities; N^T computations involving exponential time complexity. Viterbi's algorithm enables us to reduce this complexity to $O(T*N^2)$, using dynamic programming techniques. Rather than exhaustively computing all the possibilities, we maintain two data structures *alpha* (α) and *phi* (ϕ). At any point *t* in the observation sequence to be decoded, the maximum probability for emitting the observed symbol for each hidden state is stored, along with the index or argument of the maximum probable state that led there. To get the optimal state sequence we get the index of the final highest scoring hidden state and backtrack through the *phi* structure beginning with that index, returning the accumulated list. We can express this formally:

1) Initialisation: *t = 1*

$$\alpha_1(i) = \pi_i B_i(O_1) \qquad 1 \leq i \leq N$$
$$\phi_1(i) = 0$$

2) Recursion: *t = 2, …, t=T*

$$\alpha_t(j) = max_{i \varepsilon N}(\alpha_{t-1}(i)A_{ij})B_j(O_t) \quad 1 \leq j \leq N$$
$$\phi_t(j) = argmax_{i \varepsilon N}(\alpha_{t-1}(i)A_{ij}) \qquad 1 \leq i \leq N$$

3) Termination:

$$p^* = max_{i \varepsilon N}(\alpha_T(i))$$
$$S^*_T = argmax_{i \varepsilon N}(\alpha_T(i))$$

4) Backtracking: *t=T-1, …, t=1*

$$S^*_t = \phi_{t+1}(S^*_{t+1})$$

2.4 HMMs in Musical Applications

HMMs' facility for pattern recognition has been exploited for computational musical tasks. Score following, for instance, tries to consolidate the position of a live performance of a musical piece with its score representation automatically [13]. Using Viterbi decoding, an alignment can be established by extracting features for the *observed* live performance, and comparing them against idealised features within the model to return the expected location of the performance within the score.

HMMs also lend themselves quite naturally to the task of chord recognition [5, 14, 22]. Papadopoulos and Peeters [14] demonstrate a method whereby they compare the Pitch Class Profile (PCP) representation of the signal corresponding to 24 possible chord labels (12 notes for major and minor) indicated by the emission matrix, coupled with the most probable chord sequence defined in the transition matrix, derived from prior musical knowledge or training on musical scores and transcriptions.

Compared to Markov chains, HMMs have been exploited somewhat less in generative or compositional applications (apart from concatenative synthesis, which we will describe presently). Some methods however are summarised by Ferdández and Vico [6] and by Nierhaus [9], with the latter making the observation that "when applied to algorithmic composition, HMMs are appropriate to add elements to an existing composition".

3 k-Best Decoding

3.1 Parallel List Viterbi Decoder

The Viterbi algorithm has proved a robust and reliable solution in many problem applications. As we emphasise in this article however, it only outputs the maximum probability path from the model. This has been observed by other researchers as being restrictive when wanting to explore alternative paths through the system[4, 20]. Rabiner and Juang also observe, in the context of dynamic time warping, that the single solution Viterbi is often too sensitive and it is desirable to produce a "multiplicity of reasonable candidate paths so that reliable decision processing can be performed" [15]. They outline a procedure for performing k-Best decoding using what they term the Parallel Algorithm, which can be summarised as follows.

If the regular Viterbi (i.e. 1-Best) keeps track of the best scoring path leading to each state at time step t, we need to alter the algorithm and associated data structures to store the k best scoring paths that lead to that point. This necessitates expanding our *alpha* and *phi* matrices to contain $T*N*K$ probabilities and indices. We require some sorting mechanism in order to compare every possible path leading to a particular state and discarding all but the top K scoring items. We also need to keep track of a ranking to determine the order in which multiple instances of the same path arrive at a state. Depending on the size of N this sorting process can be computationally complex. The solution then, is rather than performing the sorting procedure at the end of each iteration of state $s \in N$ we keep track of all the path possibilities s in a suitable container such as a heap or priority queue.

Curiously, very few implementations exist for k-Best decoding, so we have implemented and made available ourselves a version in Python[1]. The parallel approach is so-called because at each time T it is continuously updating all k paths concurrently up to that point. In contrast, Seshadri proposes a serial algorithm that computes the k best paths sequentially, one by one [21] which is also mentioned by Rabiner and Juang [15].

3.1 *k*-Shortest Paths Graph Decoder

A HMM can also be considered as a weighted directed acyclic graph or *trellis* graph (Figure 1). For a series of T discrete observations and n hidden states, the graph contains $n*T$ nodes and n^2*T edges connecting n nodes at $t=i$ with n nodes at $t=i+1$. Edge weights correspond to the joint probability of the transition cost for each pair of nodes and the emission cost of the output symbol.

Hence the Viterbi path can be solved using shortest path algorithms such as Dijkstra's Algorithm [18]. Furthermore, there exists a number of algorithms for solving the k shortest paths and returning them as a list of the nodes ordered by cost. One of the most well-known algorithms for achieving this is Yen's Algorithm [24]. It operates by first finding the 1-best path as normal using any conventional shortest path (like

[1] https://github.com/carthach/kbestViterbi

Dijkstra). It then iterates through each node of the best path to find *spur* nodes that yield further potential shortest path candidates to the sink.

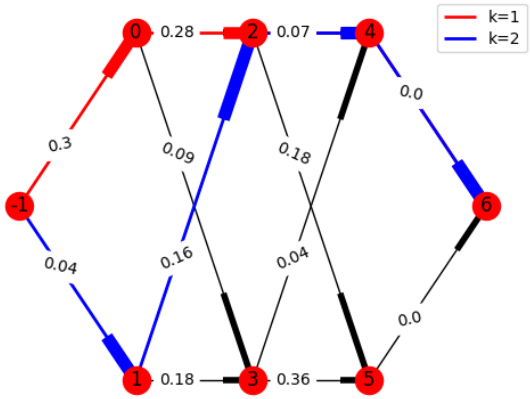


Fig. 1. Shortest Paths applied to the Wikipedia Viterbi Example[2]

Luckily an implementation of Yen's Algorithm exists in Python for the *NetworkX* graph library[3], but to convert a HMM specification to an equivalent directed acyclic graph that can be solved identically to Viterbi we need to do the following. First create $n*T$ nodes for step t of the trellis. Next create the necessary n^2*T edges and calculate the appropriate edge weights as given by $A_{ij} * B_j(O_t)$. Add a start node (-1 in the figure) and create edges connecting to each of the nodes at $t=0$ with weights encapsulating the initial probabilities $\pi_i B_i(O_1)$. Create an end node (6 in the figure) and connect it to all the nodes at $t=T$ with a zero weighting. Before we can run the shortest path algorithm we must make one final change to the edge weights for it to work. HMMs are concerned with finding maximum likelihood of multiplications of joint *probabilities* whereas shortest path algorithms determine a path through a network with minimum summations of *costs*. To address this mismatch, we must convert to negative log-space, taking care to apply the inverse afterwards to report the correct probabilities.

4 Applications in Concatenative Synthesis

4.2 Unit Selection in Concatenative Synthesis

In the context of matching a target sound or specification, unit selection solves the problem of determining what sounds to select from the corpus and the systematic structuring of the selected sounds for outputting logical concatenated sequences. Many unit selection schema have been proposed and there exists no standard or best method. However, some specific procedures have presented themselves repeatedly which we will summarise here:

[2] https://en.wikipedia.org/wiki/Viterbi_algorithm
[3] https://networkx.github.io/

Linear Search At the most basic level a linear search criterion for unit selection simply computes the (dis)similarity of every unit in the target sequence with every possible unit in the corpus, according to some distance measure (e.g. weighted Euclidean). It is a conceptually simple and robust technique. For performance sake, we have applied the linear search method ourselves in developing a real-time system for concatenative synthesis of rhythms as described in [11]. The overlying problem with this approach is that it only considers the disparity between the target and the corpus unit, and neglects treating the continuity or context of consecutive units within the output sequence.

Viterbi Decoding The Viterbi algorithm was first applied by Hunt for the purposes of speech synthesis by performing unit selection of speech phonemes samples from a prior corpus [8]. It was then adopted for musical purposes by Diemo Schwarz in his Caterpillar System [20]. By representing a unit selection system as a HMM, we can not only consider the disparity between the target and corpus unit (encoded in the emission matrix) but also the "best fit" of continuity between two consecutive units in the output sequence as determined by the transition matrix of the hidden states (Figure 2).

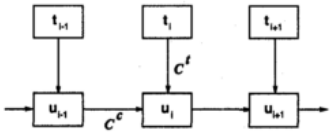


Fig. 2. Diagrammatic depiction of the relationship between target units, corpus units and their associated costs

As with shortest path problems, unit selection requires finding the minimisation of a set of weights or *costs* [8], namely the *target cost* C^t between target and corpus units and the *concatenation cost* C^c between the consecutive concatenated units [20], rather than the maximisation of probabilities. A linear combination is performed with weights w_t and w_c to give the total cost C_i. The constituent costs themselves are derived by computing the dissimilarity between the feature vectors of the associated units using a suitable distance metric, such as weighted Euclidean as is evident in Equation 1.

$$C_i = w_t {C_i}^t + w_t {C_i}^c \tag{1}$$

$${C_i}^t = \sqrt{\sum_{k=1}^{K} w^k ({t_i}^t - {u_i}^k)^2}$$

$${C_i}^c = \sqrt{\sum_{k=1}^{K} w^k ({u_i}^t - {u_{i-1}}^k)^2}$$

One of the motivations for working with more complex unit selection schemes like HMMs is that we can specify what we consider important for target cost computation separately from the concatenation cost computation by weighting accordingly or even choosing completely different features sets. For example, in speech synthesis we might choose features and weightings for the target cost that prioritise matching of length and linguistic context versus more prosodic configurations for the concatenation cost that encourage stability of energy and pitch.

Constraint Satisfaction Schwarz notes, however, that the HMM approach can be quite rigid for musical purposes because it produces one single optimised sequence without the ability to manipulate the individual units. To address these limitations, he reformulates the task into a constraint-satisfaction problem, which offers more flexibility for interaction. A constraint-satisfaction problem models a problem as a set of variables and values, and a set of constraints that allows us to identify which combinations of variables and values are violations of those constraints, thus allowing us to quickly reduce large portions of the search space [9].

Zils and Pachet first introduced constraint satisfaction for concatenative synthesis in what they describe as musical mosaicking - or, to use their portmanteau - musaicing [25]. They define two categories of constraints: segment and sequence constraints. Segment constraints control aspects of individual units (much like the target cost in a HMM-like system) based on their descriptor values. Sequence constraints apply globally and affect aspects of time, continuity, and overall distributions of units. The constraints can be applied manually by the user or learned by modeling a target. The musically tailored "adaptive search" algorithm performs a heuristic search to minimise the total global cost generated by the constraint problem. One immediate advantage of this approach over the HMM is the ability to run the algorithm several times to generate alternative sequences, whereas the Viterbi process always outputs the most optimal solution.

4.3 A *k*-Best Concatenative Synthesis System

Onset Detection and Segmentation The first stage in building a concatenative music system generally involves gathering a database of sounds to select from during the synthesis procedure. This database can be manually assembled but in many musical cases the starting point is some user-provided audio that may range in length from individual notes to phrases to complete audio tracks. In cases where the sounds destined for the sound palette exceed note or unit length, the audio needs to be separated into its constituent units using segmentation of note events analysed by onset detection, beat detection or just uniform frame lengths. For our purposes, we use onset detection of note events based on the difference of high frequency content between successive framewise spectra. Onset detection is a large topic of continuous study, and we would encourage the reader to examine the excellent review of methods summarised in [1].

Feature Extraction In order to effectively describe our audio content and produce meaningful similarity measures between constituent units both in the target sequence and the corpus of segmented units we need to choose a suitable set of descriptors. Selecting appropriate features is a trade-off between choosing the richest set capable of succinctly describing the signal, on the one hand, and the expense of storage and computational complexity, on the other.

In any case, there are a number of standard features that occur across many systems, including our own that are worth describing briefly here. We use our in-house Essentia [3] library for all our musical content analysis.

- *Loudness*: a suitable energy descriptor is essential for ensuring appropriate matching of dynamics. Essentia's descriptor is defined by Steven's power law, namely the signal's energy raised to 0.67.
- *MFCCs (Mel Frequency Cepstrum Coefficient)*: a multidimensional, compact representation of the magnitude spectrum, frequently used to make comparisons of the *timbre* of signals.
- *f0*: the fundamental frequency of the signal in question, extremely useful in tracking the predominant melody of monophonic recordings.

Unit Selection and Concatenation Extracted features for the target units and corpus units are stored in two matrices in T and C. Next, we need to compute the distance matrices $A = T * C$ *and* $E = C * C$ that will serve as our *emission* and *transition* matrices respectively. Beforehand, however, it is important to perform any normalisation or standardisation and weighting of individual features as required.

Now we can perform unit selection for any required k paths by performing the k-Best decoding function as described in Section 3.1 incorporating the changes required for handling *costs* versus probabilities extrapolated in the previous section. This function returns a multidimensional array of $k*N$ indices corresponding of length N. These indices reference individual sound units within our corpus. To produce the final audio representation, we simply concatenate the floating-point samples uniformly back to back, but there's no reason why they can't be overlapped and crossfaded at their boundaries for further smoothing as in [19]. For those interested in exploring the code it is available as a Python library on Github[4].

4 Analysis

The goal of this article is to introduce the algorithmic concepts behind k-Best decoding and a rigorous evaluation is outside the scope of this publication. Conducting such stringent surveys like we have done for previous systems [10] is always challenging, especially considering the lack of ground truth for concatenative synthesis and creative systems in general. We do, however, briefly give a glimpse of the behaviour of the algorithms in terms of performance, correctness and acoustic output here.

[4] https://github.com/carthach/PyConcat

4.1 Algorithm Performance

To compare the running times of the two implemented algorithms we took a small recording of the 8 notes of the C major scale being performing on a piano and re-synthesised it with itself. Figure 3 shows side by side of the running times for both algorithms based on the number of paths returned. Our implementation of the Parallel Decoder outperforms the NetworkX Graph approach considerably. This is mostly due the added layer of complexity of expanding the Hidden Markov Model to a fully connected directed acyclic graph. Others have noted however that NetworkX, on account of being implemented purely in Python, performs slower than other compiled graph libraries such as *graph-tool* or *igraph*.

It will be worth benchmarking against these implementations or reimplementing completely in C/C++. In any case, it is worth bearing in mind the real culprit in these applications: audio signal processing. The segmentation and analysis of the units took 0.926248 second for all runs.

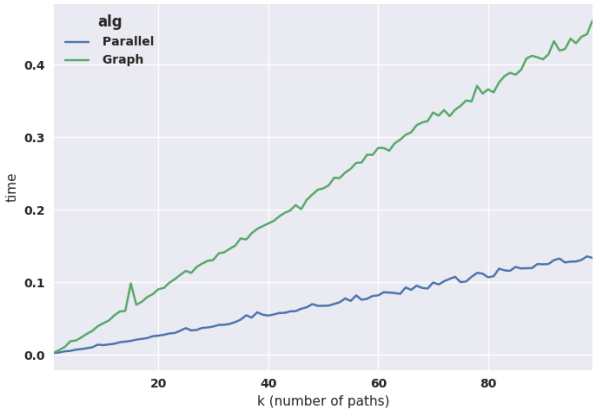


Fig. 3. Running Time Comparisons for both Algorithms

4.1 Equivalence and Correctness

Figure 4 shows the optimal sequence generated for regular Viterbi decoding and sequences $1 \leq k \leq 5$ for the Parallel Decoder and Graph Decoder respectively. The single straight line indicates that using the chosen acoustic features and weightings, the baseline Viterbi decoder correctly reassembles its own input. The Parallel and Graph decoders both correctly return this sequence as the first optimal path also (obscured by the subsequent paths in the diagram). Looking at the other returned paths, we see very slight deviations of only one unit per path (k is very small relative to the total possible paths). The possible paths, and their ordering, are identical for both algorithms.

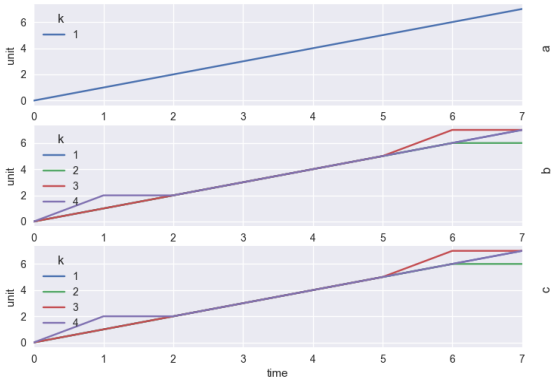
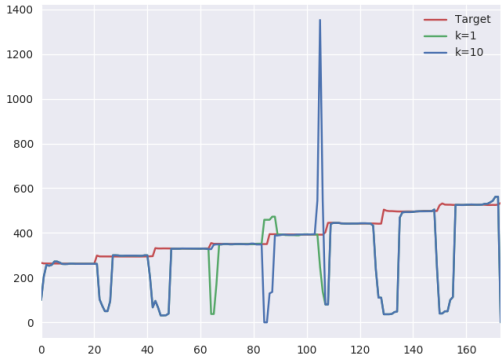


Fig. 4. Equivalency of Unit Selection Algorithms

4.2 Pitch and Timbre Preservation

To study the HMM's acoustical output, we selected energy, fundamental frequency (f0) and MFCC features, and chose a weighting scheme that gave preference to the f0 in the target cost while giving preference to the MFCCs in the concatenation cost. This allows us to focus on the specific ability of the target cost in preserving the pitch between each onset in the target sample and the selected units from the corpus, while the concatenation cost attempts to preserve a continuous and coherent evolution of timbre from a variety of timbres in the corpus. We used the same piano scale sample as a target but this time using a corpus of samples from a completely different set of instruments. From a collection of orchestral samples provided freely by the London Philharmonia[5] 610 violin, viola, clarinet and trumpet samples were gathered with notes ranging from MIDI A3 to G7.



.

Fig. 5. Plots of Fundamental Frequencies

[5] http://www.philharmonia.co.uk/explore/sound_samples

As we can see from Figure 5, and after some median filtering to remove spurious spikes, the steady states of each pitch match for generated path 1 and generated path 10 taken as examples.

Scrutinising MFCC plots is always a bit difficult to decipher, but while it is clear that the MFCC overall profile of the target (Figure 6) contrasts with the targets we can see the eight notes of the sequence reproduced and the difference in attacks that are also present in the plots of the fundamental frequencies. Again, notice lower values of k produce very similar results when the size of the corpus is large, differing only by a couple of units.

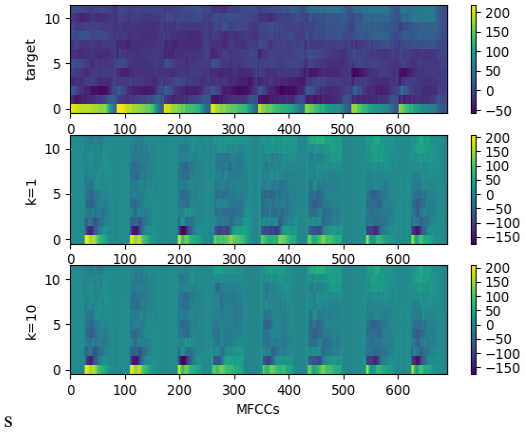


Fig. 6. MFCCs of Sequences

6 Conclusions

In this article, we described the use of the Viterbi decoding algorithm for unit selection of corpus samples in concatenative sound synthesis. One of the main criticisms of Viterbi decoding particularly in musical applications is the fact that it only outputs the single most optimal sequence. Consequently, we explained and implemented two methods for decoding the k best state sequences in a HMM using Parallel matrix decoding and k shortest paths by reformulating the problem with a graph. We then described how they can be integrated in a practical system for concatenative synthesis.

There are three directions of future work we can identify. Firstly, is the improvement of performance of the algorithm by exploring optimisation techniques and re-implementing in a compiled language. Secondly, as the results have shown, for large sized corpora there is quite low variance in lower values of k. One possible remedy for this is to choose a "stride" factor that skips every n items when gathering the k highest scoring paths at each state, thereby increasing novelty. Finally, an extensive evaluation of the system, preferably with listening surveys, is an important next stage for this research.

Acknowledgments. This research has been partially supported by the EU-funded GiantSteps project (FP7-ICT-2013-10 Grant agreement nr 610591).

References

1. Bello, J., Daudet, L.: A tutorial on onset detection in music signals. IEEE Trans. Audio. Speech. Lang. Processing. 13, 5, 1035–1047 (2005).
2. Bird, S. et al.: NLTK : The natural language toolkit NLTK : The Natural Language Toolkit. Proc. COLING/ACL Interact. Present. Sess. March, 69–72 (2016).
3. Bogdanov, D. et al.: ESSENTIA: an Audio Analysis Library for Music Information Retrieval. Int. Soc. Music Inf. Retr. Conf. (ISMIR 2013). 493–498 (2013).
4. Brown, D.G., Golod, D.: Decoding HMMs using the k best paths: algorithms and applications. BMC Bioinformatics. 11 Suppl 1, S28 (2010).
5. Cho, T. et al.: Exploring Common Variations in State of the Art Chord Recognition Systems. Sound Music Comput. 1, January, 11–22 (2010).
6. Fernández, J.D., Vico, F.: Ai methods in algorithmic composition: A comprehensive survey. J. Artif. Intell. Res. 48, 513–582 (2013).
7. Guéguen, L.: Sarment: Python modules for HMM analysis and partitioning of sequences. Bioinformatics. 21, 16, 3427–3428 (2005).
8. Hunt, A.J., Black, A.W.: Unit selection in a concatenative speech synthesis system using a large speech database. 1996 IEEE Int. Conf. Acoust. Speech, Signal Process. Conf. Proc. 1, 373–376 (1996).
9. Nierhaus, G.: Algorithmic composition: Paradigms of automated music generation. (2009).
10. Ó Nuanáin, C. et al.: An Evaluation Framework and Case Study for Rhythmic Concatenative Synthesis. In: Proc. 17th International Society for Music Information Retrieval Conference. , New York, USA (2016).
11. Ó Nuanáin, C. et al.: An Interactive Software Instrument for Real-time Rhythmic Concatenative Synthesis. In: New Interfaces for Musical Expression. , Brisbane, Australia (2016).
12. Ó Nuanáin, C. et al.: Towards User-Tailored Creative Applications of Concatenative Synthesis in Electronic Dance Music. In: MUME 2016 - The Fourth International Workshop on Musical Metacreation. , Paris, France (2016).
13. Orio, N. et al.: Score following: State of the art and new developments. Proc. Conf. New Interfaces Music. Expr. 36–41 (2003).
14. Papadopoulos, H., Peeters, G.: Large-scale study of chord estimation algorithms based on chroma representation and HMM. CBMI'2007 - 2007 Int. Work. Content-Based Multimed. Indexing, Proc. 53–60 (2007).
15. Rabiner, L., Juang, B.-H.: Fundamentals of Speech Recognition, (1993).
16. Rabiner, L.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, (1989).
17. Roads, C.: Microsound. (2004).
18. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (2nd Edition). (2002).
19. Schwarz, D.: Concatenative Sound Synthesis: The Early Years. J. New Music Res. 35, 1, 3–22 (2006).
20. Schwarz, D.: The Caterpillar System for Data-Driven Concateantive Sound Synthesis. In: Proceedings of the 6th Int. Conference on Digital Audio Effects (DAFx-03). pp. 1–6 (2003).
21. Seshadri, N., Sundberg, C.-E.W.: List Viterbi decoding algorithms with applications. IEEE Trans. Commun. 42, 2/3/4, 313–323 (1994).
22. Sheh, A., Ellis, D.P.W.: Chord segmentation and recognition using EM-trained hidden markov models. Proc. Int. Conf. Music Inf. Retr. 185–191 (2003).
23. Sturm, B.L.: Adaptive Concatenative Sound Synthesis and Its Application to Micromontage Composition. Comput. Music J. 30, 4, 46–66 (2006).
24. Yen, J.Y.: Finding the K Shortest Loopless Paths in a Network. Manage. Sci. 17, 11, 712–716 (1971).
25. Zils, A., Pachet, F.: Musical mosaicing. Digit. Audio Effects. 1–6 (2001).