SIA(M)ESE: An Algorithm for Transposition Invariant, Polyphonic Content-Based Music Retrieval

Geraint A. Wiggins
Department of Computing
City University, London
geraint@soi.city.ac.uk

Kjell Lemström
Department of Computer Science, University of Helsinki
klemstro@cs.helsinki.fi

David Meredith
Department of Computing
City University, London
dave@titanmusic.com

ABSTRACT

We introduce a novel algorithm for transposition-invariant content-based polyphonic music retrieval. Our SIA(M)ESE algorithm is capable of finding transposition invariant occurrences of a given *template*, in a database of polyphonic music called a *dataset*. We allow arbitrary *gapping*, i.e., between musical events in the dataset that have been found to match points in the template, there may be any finite number of other intervening events. SIA(M)ESE can be implemented so that it finds all transposition-invariant complete matches for a k-dimensional template of size m in a k-dimensional dataset of size n in a worst-case running time of $O(kmn + kn \log n)$; another implementation finds even the incomplete matches in $O(kmn \log(mn))$ time. The algorithm is generalizable to any arbitrary, multidimensional translation invariant pattern matching problem, where the events are representable by points in a multidimensional dataset.

1. INTRODUCTION

The SIA algorithm [5] is intended for the induction of significant structures from (musical) databases. However, it can straightforwardly be modified for use in *searching* for patterns in databases, making assessment comparisons between datasets. In the content-based music retrieval (CBMR) application, our matching algorithm, which we call SIA(M)ESE, performs to a level comparable with other, more restricted, algorithms based on, for example, dynamic programming (DP) working within an edit distance framework [2, 3, 4]. Specifying the problem as one of string matching has intrinsic drawbacks: in general, there is no natural way of representing polyphonic music exactly with linear strings (except in certain specific cases). Specifically, the string-based methods rely on linear contiguity between successive symbols in a string, and the notion of co-occurrence is somewhat vague. Either of these constraints is enough to preclude straightforward representation of polyphonic databases. However, SIA(M)ESE efficiently and effectively processes polyphonic music and produces more useful information than the DP methods. In this paper, we introduce the functionality of SIA(M)ESE and how it can be adapted to simulate other related algorithms.

2. THE NEW ALGORITHM

We express the functionality of SIA(M)ESE by using logical formulae to make it easier to understand the idea. This stage is called SIA(M), for SIA(Matching). SIA [5] is expressed by the equation (1), below. Let $\overline{v}$ represent the transposition vector. Then the structure set $\mathcal{S}$ is computed from the dataset, D, being any set of points with any number of dimensions. The dimensions may be measured on any finitely expressible metric so long as it is possible to give a total ordering, $\prec$, on all the points in the vector space defined by D.

$$\mathcal{S}(D) = \{\langle \overline{v}, X\rangle \mid (x \in X) \iff (x \in D) \wedge (x + \overline{v} \in D) \wedge (x \prec x + \overline{v})\} \quad (1)$$

The idea is to find all the maximal subsets of D, which are translated by a non-zero vector, $\overline{v}$, in the space defined by D's dimensions, to another subset of D. The ordering, $\prec$, prevents wastage of effort and duplication of results by removing repetition under symmetry. The output, S, is in the form of a set of $\langle$vector,point-set$\rangle$ pairs, relating each subset to the vector which translates it.

In SIA(M), we compute the match set $\mathcal{M}$ using (2), from a template set, T, whose occurrences in the dataset, D, we wish to find:

$$\mathcal{M} = \{\langle \overline{v}, X\rangle \mid (x \in X) \iff (x \in T) \wedge \exists y((y \in D) \wedge (x + \overline{v} \simeq y))\}. \quad (2)$$

There are just two small differences between these pairs of definitions. The more significant is that in SIA(M), the vectors, $\overline{v}$, are not specified in terms of two points, x and $x + \overline{v}$, in D ordered under $\prec$ to avoid duplication under symmetry as in (1). Rather, the vectors are generated from two separate sets of points inhabiting the same vector space: the template, T, and the dataset, D. The separation of these two sets means that the efficiency measure of removing equivalents under symmetry in generating T no longer makes sense, thus absence of the $\prec$ term of (1) in (2). The other difference is that in (2) a more relaxed notion of comparison is admitted.

In the simplest case where the aim is to find an exact occurrence of T in D, $\simeq$ in equation (2) is defined as identity and an occurrence is found if and only if condition $\exists M((\langle \overline{v}, M\rangle \in \mathcal{M}) \wedge (|M| = m))$ is also satisfied.

It turns out that the expressions above can be implemented very efficiently so that finding all transposed complete matches for a k-dimensional template of size m in a k-dimensional dataset of size n can be done in a worst-case running time of $O(kmn + kn \log n)$, another version finds even the incomplete matches in $O(kmn \log(mn))$ time. Therefore, we call the implementing algorithm SIA(M)E, for SIA(M) *Express*.

3. FINDING MATCHING DATA

To reason about the structures produced by the SIA(M)E algorithm to generate descriptions of matches with particular properties between a template and a dataset, we introduce the notion of a *cover*. This allows us formally to describe the corresponding elements in a match, and some useful properties of covers which can be used to determine particular kinds of match. The process works by selecting covers of a template T from $\mathcal{M}$ under a heuristic evaluation function composed from various possible elements. Since we are now adding Selection and Evaluation to SIA(M)E, we call this completed algorithm SIA(M)ESE.

3.1 Pattern Cover for a Matching Subset

Having computed $\mathcal{M}$, we search in it for a cover of T. The intuition behind the cover is that we want an optimal (in some appropriate sense) set of fragmentary matches to our template, T, such that each datapoint in each of (the largest possible subset of) T and the

matching subset of $\mathcal{M}$ is accounted for exactly once. So a cover describes a one-to-one mapping between (the largest possible subset of) T and that subset of $\mathcal{M}$ against which T has been matched.

Stated more formally, a candidate cover C is a set of ⟨vector,datapoint-set⟩ pairs (*viz.*, a subset of $\mathcal{M}$), the intersection of whose datapoint-sets is empty, and none of whose datapoint-sets contains a datapoint which maps, under the translation of its associated vector, to the same point as a member of a different datapoint-set in C under the translation of its own respective vector:

$$\begin{aligned} & ((C \subseteq \mathcal{M}) \wedge \forall \overline{v}_1, \overline{v}_2, X_1, X_2(\langle \overline{v}_1, X_1 \rangle \in C) \wedge \\ & (\langle \overline{v}_2, X_2 \rangle \in C) \wedge (\overline{v}_1 \neq \overline{v}_2)) \\ \rightarrow \quad & \neg \exists x_1, x_2((x_1 \in X_1) \wedge (x_2 \in X_2) \wedge ((x_1 = x_2) \vee \\ & (x_1 + \overline{v}_1 = x_2 + \overline{v}_2))) \end{aligned} \tag{3}$$

The *cover*, $\mathcal{C}$, is the set of such pairs the union of whose datapoint-sets is the largest out of the candidate covers.

However, (3) will typically generate many covers, with different structural properties. We need to be able to pick an appropriate one. Often, we will want to use the *smallest cover* – that is, the one which divides the pattern up into fewest segments. This is because we are working under the assumption that our data will be coherent where it does not contain errors, and because we want to find the closest match in the case where there is not a unique solution. In general, it will be infeasible to find the absolutely smallest cover. Instead, we use an algorithm which is a straightforward specialization of best first search. In coherent cases, this technique is efficient, even though it is search-based, as the data subsets will be large, and so it will be relatively easy to assemble a complete match.

Inferring Matches and Differences. However, there may be several smallest covers, and they may have different properties which may be useful in different circumstances. In the context of CBMR, we would want the sets in our cover to match (almost) time-contiguous sets in the data, so that they constitute a complete musical unit. We call these the time-minimal and the time-maximal smallest covers, respectively. Note that while these and other data-dependent properties might be built in to the basic algorithms for SIA(M) (and SIA) we have chosen to keep them separate from the symmetrical multidimensional specification for reasons of clarity and neatness.

4. SIMULATING RELATED ALGORITHMS

In the literature, there are only a few CBMR algorithms capable of matching polyphonic data. In certain cases, some of those may be more efficient than SIA(M)E (as, e.g., MONOPOLY in finding exact occurrences of a monophonic template in polyphonic dataset). However, we claim that none of them is as flexible and versatile CBMR algorithm as SIA(M)E. In the following, we illustrate this by showing how the working of the previous CBMR algorithms can be simulated by using SIA(M)E (without the need of changing its original time complexity). This is not the case the other way around: none of the previous algorithms can simulate the working of SIA(M)E, at least, not without major modification leading to a considerable increase of time complexity.

Gapping Simulation. Even though it is an advantage to be able to deal with unlimited gaps, if desired, we can simulate the gapping used in [1, 2] with SIA(M)E. In this case, we use a mechanism that we call *labelling*. We define a label to be an attribute attached to every data point, but which is not being considered as another dimension by SIA(M)E. Thus, we are able to represent dimensions not exhibiting translation invariance.

In this case, SIA(M)E attaches an extra label, $K(p)$, for each two-dimensional point p in the dataset (the labelling mechanism is naturally generalizable to any n-dimensional dataset). Let $ot(p)$ denote the *onset time* of datapoint p. The mapping K is a surjection to a range $0, \ldots, n'$, such that $K(p_1) > K(p_2)$ if and only if the onset time of data point p_1 occurs later than that of p_2, and $K(p_1) = K(p_2)$ if and only if $ot(p_1) = ot(p_2)$. Moreover, if p_1 and p_2 are two datapoints such that $ot(p_1) < ot(p_2)$ and there exists no datapoint p such that $ot(p_1) < ot(p) < ot(p_2)$ then $K(p_2) = K(p_1) + 1$.

Now, the modification to SIA(M)E is slight. Recall the formulae above. Having calculated equations (2), instead of observing if there is an M in $\mathcal{M}$ such that $|M| = m$, the truth of the following sentence should be observed:

$$\begin{aligned} \exists X \qquad & ((\langle \overline{v}, X \rangle \in \mathcal{M}) \wedge (|X| = m) \\ \bigwedge_{p_1 \in \tau(X, \overline{v})} \quad & \exists p_2((p_2 \in X) \wedge (|K(p_2) - K(p_1)| \leq t))) \end{aligned} \tag{4}$$

where $\tau(X, \overline{v})$ denotes the set of datapoints in the dataset that is equal to the translation of X by $\overline{v}$ and t is the allowable gap. Whereas the allowing of a parametrized spacing makes the string matching approach more complex, for SIA(M)E it does not have such an affect; it causes only a minor practical overhead for the execution.

Multiple Simultaneous Templates. Simultaneous searching for multiple musical templates, considered in [3], can also be solved with SIA(M)E. Let $T_1 \cdots T_h$ be the h templates to be searched for in D, simultaneously. With SIA(M)E, the solution is very straightforward; it needs no modification to the problem specification nor to the implementation. We simply consider the numbering of the templates as another dimension to the problem. Because the points in the dataset have a constant value along this dimension, it is not possible for any two points p_i and p_j of two distinct templates to be translatable with the same vector.

5. CONCLUSION

We have a introduced the functionality of a novel efficient and versatile transposition invariant CBMR algorithm for polyphonic music. Indeed, SIA(M)ESE is generalizable to any arbitrary, multidimensional translation invariant pattern matching problem, where the events can be represented by points in a multidimensional dataset. Thus, e.g., it is also applicable in matching bit-map images. All the details of SIA(M)ESE will appear in a complete manuscript.

6. ACKNOWLEDGEMENTS

This work was partly supported by grants #48313 from the Academy of Finland and GR/R25316 from EPSRC (Kjell Lemström), and EPSRC research grant GR/N08049 (David Meredith). Patents applied for [6].

7. REFERENCES

[1] M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and W. Rytter. Finding motifs with gaps. In *Proc. ISMIR'2000*, Plymouth.

[2] M. Dovey. A technique for regular expression style searching in polyphonic music. In *Proc. ISMIR'2001*, Bloomington.

[3] J. Holub, C. Iliopoulos, and L. Mouchard. Distributed string matching using finite automata. *Journal of Automata, Languages and Combinatorics*, 6(2):191–204, 2001.

[4] K. Lemström and J. Tarhio. Detecting monophonic patterns within polyphonic sources. In *Content-Based Multimedia Information Access Conference Proceedings (RIAO'2000)*, pages 1261–1279, Paris, 2000.

[5] D. Meredith, K. Lemström, and G. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 2002. (To appear).

[6] D. Meredith, G. Wiggins, and K. Lemström. Method of pattern discovery, 2002. PCT patent application number PCT/GB02/02430, UK patent application number 0211914.7.