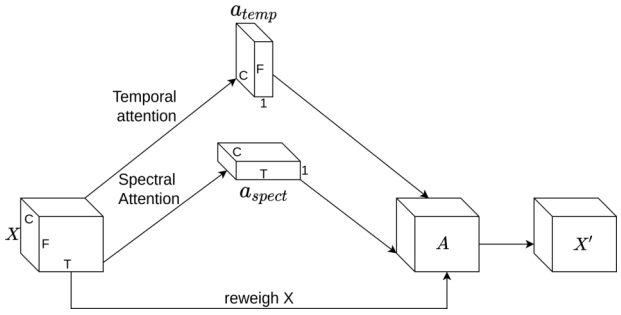

Figure 1. The illustration of spectral-temporal attention mechanism on input feature.

segments. Furthermore, the back-end consists of subsequent resnet blocks containing 2 convolutional layers. The spatial dimensions of each block are downsampled using a stride of 2x2, while the depth of each block is designed to be double of the preceding layer. Note that every convolution layer uses kernels of size 3x3, and a ReLU activation and batch normalization follows each convolution layer. The overall structure of the architecture is presented in Table 1.

CNNs have demonstrated their ability to extract complex features from low-level features, such as the log Mel spectrogram. However, the CNN features are translation-invariant, which implies that the spatial regions are treated equally in the feature map, which may not be useful in the audio context. Therefore, using the attention mechanism, we aim to enhance the interesting spatial patches by assigning more weight to time indices and frequency bands containing salient information and vice versa.

In order to apply the spectral-temporal attention mechanism, a channel-wise mask is computed and applied to the CNN features X of dimension $C \times F \times T$. We use two different attention weights, denoted by $a^{temp} \in R^{C\times F\times 1}$ for temporal attention and $a^{spect} \in R^{C\times 1\times T}$ for spectral attention that are modeled as:

$$\boldsymbol{a^{temp}} = \mathrm{softmax}(\boldsymbol{X^T W_{temp}}), \tag{1}$$

$$\boldsymbol{a^{spect}} = \mathrm{softmax}(\boldsymbol{X^T W_{spect}}), \tag{2}$$

where $\boldsymbol{W_{temp}}$ and $\boldsymbol{W_{spect}}$ are learnable weights. Furthermore, the spectral-temporal attention mask $\boldsymbol{A} \in R^{C\times F\times T}$ is computed using the outer product between attention weights:

$$\boldsymbol{A} = \boldsymbol{a^{spect}} \otimes \boldsymbol{a^{temp}} \times S \tag{3}$$

where, S is a scaling factor to rescale the attention mask to enable easy gradient flow during model training. Finally, the CNN feature map is reweighted using the mask $\boldsymbol{A}$ as:

$$\boldsymbol{X'} = \boldsymbol{A} * \boldsymbol{X} \tag{4}$$

2.3.2 Projection Head

The final output of the CNN encoder is projected into a lower-dimensional latent space via a projection head. The fully connected layers in the projection head result in a large number of model weights. Therefore, as used in previous studies [9, 16], we utilize the split head to split the encoder output into d branches. Each branch is then passed through the corresponding linear layers and their final outputs are finally concatenated to generate a d-dimensional embedding, followed by L^2 normalization.

2.4 Contrastive Learning framework

We employ a contrastive learning approach [14] for training the model to map the similar audio samples closer together and pull away different audio samples. As discussed in Section 2.2, pairs of clean segments and their corresponding distorted versions are generated for training the model. These pairs are equivalent to the anchor and positive sample pairs as denoted in the simCLR framework. For each sample in a mini-batch of size N (N/2 pairs), there are N-2 negative samples since we also allow a positive sample to be an anchor sample in the batch. We train the proposed model using these pairs that are mapped into the latent space where the contrastive loss is formulated. We use the normalized temperature-scaled cross-entropy loss as devised in [14]:

$$\mathcal{L}_{\mathrm{i,j}} = -\log \frac{\exp(\mathrm{sim}(\mathrm{e_i}, \mathrm{e_j})/\tau)}{\sum_{\mathrm{k}=1}^{\mathrm{N}-1} 1_{[\mathrm{k}\neq\mathrm{i}]} \exp(\mathrm{sim}(\mathrm{e_i}, \mathrm{e_k})/\tau)}, \tag{5}$$

where, τ is a temperature hyperparameter that allows learning from hard negatives. The cosine similarity is used to calculate pairwise similarity.

2.5 Database-creation and Indexing

The trained model is used as a fingerprinter to extract fingerprints from the audio tracks. The fingerprint for a given audio track is generated as follows: First, we extract segments of length L with a hop size of H. These segments are pre-processed as mentioned in Section 2.1. Then, the audio segments are transformed into log Mel spectrograms and fed to the model to generate encoded embeddings. We term these embeddings as subfingerprints, which form a fingerprint of the entire audio track when stacked in the time–ordering sequence. Finally, we extract the subfingerprints of every audio segment in the database and link them with their corresponding timestamps and the audio identifier to create a reference database.

Finding the nearest neighbors for a sample in a d-dimensional space is not a trivial task (when *d>50*) [6]. Also, the brute-force search strategy in a massive database is computationally expensive. Therefore, we employ the Locality Sensitive Hashing (LSH) [17] to index the subfingerprints database. It allows for efficient retrieval by only comparing a fraction of the dataset to retrieve the exact match. Moreover, it adds the noise-robustness property to the retrieval process.

2.6 Retrieval process

This section describes the process of finding the closest matching audio snippet from an indexed database, given a

query audio snippet. For a given query audio, we follow the same procedure to generate subfingerprints as mentioned in the previous subsection. Suppose the query $Q = (q_m)_{m=1}^{M}$ is composed of M subfingerprints. We use LSH to retrieve the nearest match for each subfingerprint q_m, denoted as I_m (the index). To identify the closest matching audio track, we retain counts of the audio identifiers corresponding to the retrieved matches. The query audio is finally identified with the audio identifier with the maximum counts.

We also locate the precise timestamp of the query audio in the identified audio track. To accomplish this, we first eliminate the retrieved matches that do not belong to the identified audio track. Next, we generate the set of candidate sequences S_i of length M with their starting indices as $I_i = I_m - m$. Finally, we select candidate sequence S_i, which have at least 50% intersecting retrieved indices. The timestamp corresponding to I_i is chosen as the matching timestamp of query audio in the retrieved audio track.

3. EXPERIMENTAL SETUP

3.1 Dataset

- **Free Music Archival (FMA)** [18]: is an open, large-scale, and easily accessible dataset commonly used for various music information retrieval tasks. We used the *fma_large* and *fma_medium* versions of the dataset for model training and testing, respectively. There are 106k and 25k 30s audio clips in the *fma_large* and *fma_medium*, respectively. Note that we excluded the shared clips between both datasets for training the model.
- **Noises:** We used a range of real-world background noises for training and testing. We extracted the noise signals from database [1] , which includes a wide range of sounds from the MUSAN corpus [19], for training the model. For testing the system, we used the ETSI database [2] , from which we chose seven distinct background noises: Babble, Living Room, Cafeteria, Car, Workplace, Traffic, and Train Station.
- **Room Impulse Responses (RIRs):** We used real RIRs [1] corresponding to different acoustic environments, ranging from a small room to a large hall, for training the model. We selected six RIRs from Aachen Impulse Response Database [20] with t60 values ranging from 0.1s to 0.8s for testing the system.

3.2 Implementation details

We compared our approach with a baseline system [10] that generates fingerprints using an encoder similar to Now-Playing's [9] architecture and does a comprehensive search. To the best of our knowledge, it is the only method that employs a neural network model to generate robust audio fingerprints and performs better than conventional approaches. We chose the log Mel spectrograms as input to our model and the baseline method. Table 2 lists the experiments configurations used for developing our audio fingerprinting system.

Parameter	Value
Sampling rate (F_s)	16 kHz
Audio segment length (L)	960 ms
Energy threshold (t)	0 dB
log Mel spectrogram dimensions (F $\times$ T)	64 $\times$ 96
Subfingerprint hop length (H)	100 ms
Subfingerprint dimensions (d)	128
Batch size (N)	512
Scaling factor (S)	100
LSH configuration:	
Tables	50
Hash bits	18
Number of probes	200

Table 2. Experiments configurations

We trained the models with the Adam [21] optimizer for 150 epochs using the cyclic learning rate. The initial learning rate was 5e-4 and reached a maximum of 5e-2 in 40 epochs. We also tweaked the temperature hyperparameter in the [0.01-0.1] range and found no significant benefits. The model was trained on a single NVIDIA Tesla V100 GPU for about 40 hours.

We built a reference database of $\sim$7.3M fingerprints using the *fma_medium* dataset. We used LSH implementation [3] to index the generated audio subfingerprints. Furthermore, the retrieval performance was equivalent to the brute force search after fine-tuning the LSH, with less than a 0.1% drop in retrieval accuracy at various distortion levels.

3.3 Evaluation metric

We used the following metric for system evaluation at audio/segment level retrieval:

$$accuracy = \frac{n\ hits\ @\ top\text{-}1}{n\ hits\ @\ top\text{-}1 + n\ miss\ @\ top\text{-}1} \times 100 \quad (6)$$

Note that a match is declared correct for the segment level search if the located timestamp of the query in the correct retrieved audio is within $\pm$ 50 ms.

3.4 Search query

We used the *fma_medium* dataset to generate 10,000 search queries, which were randomly extracted from different audio tracks. To assess system performance at different distortion levels, we distorted queries with noise, reverberation, and a combination of both. To generate a noisy reverberant audio query, we first convolved both the audio and the noise signal with the RIR corresponding to a t60 level

[1] https://www.openslr.org/28/
[2] https://docbox.etsi.org/
[3] https://github.com/FALCONN-LIB/FALCONN

of 0.5s and then added both signals at SNR levels ranging from 0dB to 25dB. In addition, we created queries of lengths: 1s, 2s 3s, and 5s to test our system efficiency for varying lengths.

4. RESULTS AND DISCUSSION

In the following, we present the performance of our system under different distortion conditions. We compare our system with the baseline system as mentioned in the previous section and the Audfprint system.

4.1 VS. Baseline system

- **Noise:** Table 3 presents the performance of the systems in noisy conditions. It can be seen that our system performs better than the baseline system by a reasonable margin, particularly at the 0dB and 5dB SNR levels. Moreover, our system can precisely locate the timestamp with reasonable accuracy, given enough query length (>2s) in very high noise conditions too. The performance gap narrows with the increase in SNR level and becomes smaller from 20dB onwards. We noted that the performance gap between systems was less than 2% at SNR levels of 20dB and more, irrespective of the query lengths.

- **Reverb:** As can be seen in Table 4, the retrieval accuracy of the systems drops with an increase in t60 levels, i.e., high reverberant environments. Furthermore, we discovered that the reverberation causes embeddings correspondings to adjacent audio segments to be very similar, which is most likely the reason that the correct match was not found at the top-1 rank in many cases, which resulted in low retrieval performance of the system. Nevertheless, the presented results indicate that our system performs effectively even in high reverberation environments for short query snippets. The baseline system is quite effective in a low reverberation environment with more than 80% retrieval accuracy. However, its performance falls short at higher reverberations, even with long query snippets.

- **Noise and Reverb:** The systems were tested in a more challenging situation by considering a noisy and reverberant environment. As shown in Table 5, the performance of both systems degrades with the addition of reverberation compared to their performance under noisy conditions. Due to added reverberation, the retrieval accuracy of our system drops by around 15% at 0dB SNR, but it improves on the less noisy conditions. On the contrary, the baseline system performs poorly, particularly at 0dB SNR. Furthermore, the accuracy gap remains over 15% compared to its performance in noisy conditions, even at higher SNR levels. It indicates that our system is more resilient against noisy and reverberant environments than the baseline system.

Method	Query length (s)	0dB	5dB	10dB	15dB
Ours	0.96	**76.8**	**84.8**	**87.4**	**88.5**
Baseline		62.7	79.4	85.5	87.6
Ours	2	**82.7**	**90.8**	**93.2**	**94.1**
Baseline		71.1	86.5	90.4	92.0
Ours	3	**83.9**	**92.9**	**94.6**	**95.5**
Baseline		76.8	88.5	91.4	93.8
Ours	5	**85.8**	**93.9**	**94.7**	**96.8**
Baseline		79.8	89.4	91.5	94.2

Table 3. Top-1 hit rate (%) performance in the segment-level search for varying query lengths in noisy conditions.

Method	Query length (s)	0.2s	0.4s	0.5s	0.7s	0.8s
Ours	0.96	**85.1**	**84.1**	**78.8**	**83.3**	**74.4**
Baseline		78.4	75.5	67.5	75.1	62.2
Ours	2	**89.1**	**87.3**	**80.6**	**85.4**	**75.0**
Baseline		85.5	81.3	72.5	78.6	64.9
Ours	3	**90.1**	**87.9**	**81.0**	**86.8**	**76.3**
Baseline		87.2	83.3	73.9	80.8	66.9
Ours	5	**91.2**	**89.6**	**82.6**	**88.4**	**77.4**
Baseline		87.8	84.3	75.1	81.9	67.6

Table 4. Top-1 hit rate (%) performance in the segment-level search for varying query lengths in reverberant conditions.

The above-stated results show that our system performs reasonably well with short query snippets in different distortion environments. Furthermore, it indicates that our system does not require long queries to achieve reasonable performance at higher distortion levels. However, the performance of the systems improves with the increase in the query length using our proposed simple yet effective sequence search strategy. Therefore, the system can be fed with longer queries to obtain more reliable results.

Attention mechanism effect: Based on system performance in a noisy reverberant environment, we examine the effectiveness of adding an attention mechanism to the model. Table 6 shows that the attention mechanism improves retrieval accuracy, particularly at low SNR levels, with small benefits at higher SNR levels. Furthermore, we noted similar results hold true in other distortion conditions, i.e., noisy and reverberant environments. These results support that the spatial-temporal attention mechanism enhances the CNN to generate robust audio embeddings.

Embedding dimensions: We examined the effect of the audio embedding dimension on the system performance using the 0.96s queries. We observe that shrinking the dimensions from 128 to 64 has little impact on the system, especially in the reverberant environment. However, in the noisy reverberant and noisy environments, retrieval

Method	Query length(s)	0dB	5dB	10dB	15dB
Ours	0.96	**60.3**	**76.6**	**81.3**	**82.8**
Baseline		27.3	58.7	70.7	73.9
Ours	2	**66.4**	**83.5**	**86.9**	**88.0**
Baseline		39.0	69.6	76.5	78.7
Ours	3	**67.9**	**85.1**	**88.2**	**89.3**
Baseline		47.1	75.2	80.2	81.4
Ours	5	**69.5**	**87.1**	**90.5**	**91.9**
Baseline		54.7	77.3	81.8	82.8

Table 5. Top-1 hit rate (%) performance in the segment-level search for varying query lengths in noisy reverberant conditions.

	0dB	5dB	10dB	15dB	20dB
Attention	60.3	76.6	81.3	82.8	84.6
No attention	52.4	68.1	75.7	79.9	82.3

Table 6. The effect of added attention mechanism on Top1-hit rate performance of the system in noisy reverberant environments for 0.96s long queries.

accuracy declines by 5.6% and 3.8% at 0dB SNR, respectively, whereas accuracy losses are minimal at other SNR levels. Furthermore, increasing the number of dimensions from 128 to 256 does not provide significant improvements. This investigation allows reducing dimensions further without a significant performance drop to resolve the space constraints problem.

Computational and Memory load: We further investigated the efficiency of our system based on its computational and memory requirements. The final size of the sub-fingerprints database is roughly 1.25GB, with 128 32-bit floating numbers representing each audio subfingerprint. We use the *Intel Xeon Platinum 8268* CPU to do an in-memory search due to the small database size. We report that the LSH takes about 0.01s to retrieve top-5 matches for a query subfingerprint. Moreover, our system takes about 0.25s to process a 3s long query and locate its timestamp in the identified reference audio. It is also worth noting that the retrieval process can be sped up by employing a parallel search method. Furthermore, there is scope for investigating the computation load reduction by storing subfingerprints in low-bit floating numbers without a significant drop in the retrieval accuracy.

4.2 VS. Audfprint

We also compare our system with the Audfprint [4] based on the Shazam method. We notice that Audfprint performs poorly in the segment level search, particularly at high distortion levels. Therefore, we only compare its performance with ours for the audio identification task. Moreover, its performance deteriorates with short query lengths; hence we present the retrieval results with 5s audio query snippets in Table 7.

It can be seen that our system delivers excellent retrieval accuracy, with over 95% accuracy under high distortion conditions. On the contrary, the Audfprint method performance degrades severely at high distortion levels, which indicates that our system also outperforms the conventional approach for the audio identification task. It should be noted that the Audfprint system uses a hash table to index the database, resulting in reduced fingerprint database size. Furthermore, the size of the database generated by Audfprint is around 400 MB, which is roughly three times less than ours.

Distortion	Method	0dB	5dB	10dB	15dB
Noise	Ours	**95.0**	**98.7**	**98.9**	**99.2**
	Audfprint	72.1	82.7	89.4	91.2
Noise+ Reverb	Ours	**84.3**	**96.8**	**98.5**	**98.9**
	Audfprint	64.8	79.4	87.2	92.3
		0.2s	0.4s	0.5s	0.7s	0.8s
Reverb	Ours	**99.2**	**99.5**	**98.9**	**99.6**	**98.7**
	Audfprint	96.1	94.6	81.8	89.6	40.2

Table 7. Top-1 hit rate (%) performance in the audio-level search for varying lengths in different distortion conditions.

5. CONCLUSION

This paper presents an audio fingerprinting system robust against high noise and reverberation conditions. Our work focuses on generating robust audio embeddings by employing a contrastive learning framework. Moreover, we propose to enhance CNN with the channel-wise spectral-temporal attention mechanism to reweigh the CNN features. This enables CNN to assign more weight to salient patches in the CNN features, resulting in discriminative audio embeddings. Furthermore, our system performs a comprehensive search to precisely estimate the timestamp of the query in the identified reference audio using a simple sequence search strategy, which makes our system applicable to audio synchronization tasks. Our system performs well compared to the baseline [10] and Audfprint [4] methods. Also, our system is computationally and memory-efficient due to the compact embeddings that make our system deployable on an extensive database. The future direction of this work is to obtain discrete audio embeddings to speed up the audio retrieval process.

6. ACKNOWLEGMENT

This work has been supported by the research grant(PB/EE/2021128-B) from Prasar Bharati. We would also like to extend our gratitude to IITK Paramsanganak for their GPU computing resources.

[4] https://github.com/dpwe/audfprint

7. REFERENCES

[1] "Shazam music recognition service," http://www.shazam.com/.

[2] "Soundhound," http://www.soundhound.com/.

[3] C. Howson, E. Gautier, P. Gilberton, A. Laurent, and Y. Legallais, "Second screen tv synchronization," in *2011 IEEE International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 2011, pp. 361–365.

[4] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[5] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system." in *Ismir*, vol. 2002, 2002, pp. 107–115.

[6] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern recognition*, vol. 41, no. 11, pp. 3467–3480, 2008.

[7] T. Shibuya, M. Abe, and M. Nishiguchi, "Audio fingerprinting robust against reverberation and noise based on quantification of sinusoidality," in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, pp. 1–6.

[8] A. Báez-Suárez, N. Shah, J. A. Nolazco-Flores, S.-H. S. Huang, O. Gnawali, and W. Shi, "Samaf: Sequence-to-sequence autoencoder model for audio fingerprinting," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2, pp. 1–23, 2020.

[9] B. Gfeller, B. Aguera-Arcas, D. Roblek, J. D. Lyon, J. J. Odell, K. Kilgour, M. Ritter, M. Sharifi, M. Velimirović, R. Guo, and S. Kumar, "Now playing: Continuous low-power music recognition," in *NIPS 2017 Workshop: Machine Learning on the Phone*, 2017.

[10] S. Chang, D. Lee, J. Park, H. Lim, K. Lee, K. Ko, and Y. Han, "Neural audio fingerprint for high-specific audio retrieval based on contrastive learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3025–3029.

[11] E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[12] H. Wang, Y. Zou, D. Chong, and W. Wang, "Environmental Sound Classification with Parallel Temporal-Spectral Attention," in *Proc. Interspeech 2020*, 2020, pp. 821–825.

[13] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sep. 2015, pp. 1412–1421.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3270–3278.

[17] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 253–262.

[18] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2016.

[19] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[20] M. Jeub, M. Schäfer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *Proceedings of International Conference on Digital Signal Processing (DSP)*, IEEE, IET, EURASIP. Santorini, Greece: IEEE, Jul. 2009, pp. 1–4.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

SIATEC-C: COMPUTATIONALLY EFFICIENT REPEATED PATTERN DISCOVERY IN POLYPHONIC MUSIC

Otso Björklund
University of Helsinki

ABSTRACT

The use of point-set representations of music enable repeated pattern discovery to be performed on polyphonic music. The discovery of patterns containing polyphony is also enabled by the use of point-set representations. The SIA and SIATEC algorithms discover repeated patterns in point-sets by computing maximal translatable patterns and their translational equivalence classes. While the algorithms are relatively efficient, their application to larger pieces of music is not viable due to quadratic space complexity. This paper introcudes a novel algorithm, SIATEC-C, for repeated pattern discovery in point-set representations of music. The algorithm discovers repeated patterns and finds all of their occurrences, while running with subquadratic space complexity. The algorithm can also provide significant running time improvements over the comparable SIATEC algorithm. The computational performance of the algorithm is compared with SIATEC. The accuracy of the algorithm is also evaluated on the JKU-PDD data set.

1. INTRODUCTION

This paper presents a novel algorithm for computationally efficient repeated pattern discovery in symbolically represented polyphonic music. The main contribution of the algorithm is efficient production of candidate patterns for a post-processing phase where the patterns can be refined, and musically unimportant patterns can be filtered out.

The goal of repeated *pattern discovery* is to find musically important patterns and their occurrences in a piece of music (*intra-opus*) or in a corpus (*inter-opus*). In repeated pattern discovery no query is given by the user unlike in *pattern matching*, where the goal is to find occurrences of a query pattern [1]. Decomposing a piece of music into its constituent elements is a fundamental part of music analysis [2]. Repeated pattern discovery can thus be applied to multiple problems in computational music analysis, such as motivic analysis [3, 4], tune classification [5], segmentation [6], and style analysis [7]. A compressed representation of a piece formed by its repeated patterns can even be considered an analysis of the piece [8, 9].

Repeated pattern discovery is a challenging problem due to certain characteristics of music. Repetition is prevalent in music causing algorithms that focus just on finding repetitions to output large numbers of patterns even for short pieces of music [4, 10]. Often most of the repeated patterns discovered by an algorithm do not correspond to patterns that have musical significance [11]. Instead of relying on repetition alone for discovering musically significant pattern, methods for identifying which patterns are musically important are needed.

The SIATEC-C algorithm presented in this paper improves upon the running time and space complexity of previous point-set algorithms by avoiding the computation of patterns with large temporal gaps and small patterns that are already included in a larger pattern.

2. RELATED WORK

String representations of music have been employed for both pattern matching and discovery in monophonic music. Monophonic music can be represented as a string for pattern matching [12] or a set of strings for pattern discovery [13]. String representations have been often used for mining *closed* patterns, that is, patterns that cannot be extended without reducing the number of occurrences [4, 10]. Monophonic music can also be represented as a pitch signal which enables the use of signal processing methods. Wavelet analysis is used by [14] for repeated pattern discovery in monophonic music.

Polyphonic music can be split into monophonic voices in order to use monophonic pattern discovery methods (e.g., in [13, 15]. Voice separation can be challenging if the voice information is not included, and using monophonic pattern discovery on separate voices cannot be directly used to discover polyphonic patterns or patterns that move from one voice to another, such as *call-and-response* patterns in jazz.

Point-set representations of music enable pattern matching [16] and discovery in polyphonic music [11] with patterns that can be polyphonic. Time-shifts and transpositions of a pattern can be performed by translating the points of the pattern by a vector. The SIA-family of algorithms (see [17] for an overview) discover repeated patterns in music by computing *maximal translatable patterns* (MTP) and finding their occurrences by computing the *translational equivalence classes* (TEC) for the patterns ([11]). The definitions of MTPs and TECs are covered in section 3.

The SIA algorithm by [11] computes all MTPs in a k-dimensional point-set of n points in $O(kn^2 \log n)$ worst-case time and $O(kn^2)$ space, and the SIATEC algorithm computes the TECs of all MTPs in $O(kn^3)$ worst-case time and $O(kn^2)$ space. The worst-case space complexity of computing all MTPs has been improved upon by [18] to $O(kn)$. Another variant of SIA is the SIAR algorithm by [19], which aims to improve the computational performance of MTP computation by restricting the number of difference vectors by a sliding window.

A musical pattern may be *time-warped* and *time-scaled*. [20] present an algorithm that can discover transposed and time-warped repeating patterns in a two-dimensional point-set of n points in $O(n^2 \log n)$ time. The algorithm by [21] can discover transposed and time-scaled repeated patterns in a two-dimensional point-set in $O(n^4 \log n)$ time.

The number of patterns discovered by computing MTPs can often be very large even for small point-sets [11]. Various approaches based on the use of *compression ratio*, *compactness*, and other heuristics have been developed for selecting the musically most important patterns from the set of discovered MTPs. The COSIATEC and SIATEC-Compress algorithms [9, 22] compute a compressed representation of a point-set. The algorithm by [23] similarly aims to compute a representation formed by musically important patterns. [24] further develops the idea of computing a highly compressed representation of a point-set by recursively splitting the input point-set and refining the TECs by removing redundant translation vectors.

MTPs can contain large temporal gaps, such that, the MTP may consist of a musically important pattern and additional points called *isolated members* [19, 25]. [19] proposes a solution to the problem of isolated membership by using an additional processing stage called *compactness trawling* to split MTPs into smaller patterns without isolated members. Post-processing MTPs with compactness trawling in SIA has been found to improve precision and recall [26] over plain SIA. Compactness trawling combined with the symbolic fingerprinting method presented in [27] have been employed in the SIARCT-CFP algorithm to discover inexact occurrences of repeated patterns with high precision and recall [28].

3. BACKGROUND AND DEFINITIONS

In a point-set representation of music, note events are represented as points and a piece of music is represented as a point-set $D \subset \mathbb{R}^k$, where k is the dimensionality of the points. Often $k = 2$, where the first dimension represents the onset time of a note event, and the second dimension represents the pitch of the note event. The rest of this paper assumes that the points are two-dimensional, and $p.x$ denotes the onset time and $p.y$ the pitch of a point p. The number of points in a point-set D is denoted by $|D|$ or n. By using only the onset time of the note events, patterns can be matched based on the *inter-onset-intervals* (IOI) of the adjacent notes. This allows for variations in the durations of the notes. The IOI between consecutive note events p_1 and p_2 is defined as the difference $p_2.x - p_1.x$.

A point-set can be sorted by using a *lexicographical ordering* [11]. A two-dimensional point p_1 is considered to be less than a point p_2, if and only if, $p_1.x < p_2.x$ or $p_1.x = p_2.x$ and $p_1.y < p_2.y$. Lexicographical ordering can be extended to points of any dimensionality. A lexicographically sorted point-set is denoted D_s.

A pattern $P \subset \mathbb{R}^k$ is also a point-set, and P is said to occur in D if $P \subseteq D$. Shifting a pattern P in time and transposing it can be expressed as translation by a vector t, where $t.x$ is the time shift and $t.y$ is the transposition. Translating a pattern P by a vector t is denoted $P + t$ [11]. The points in patterns are assumed to be in ascending lexicographical order in this paper.

The maximal translatable pattern, ***MTP***, of a vector t in a point-set D is defined by [11] as

$$MTP(t, D) = \{d \mid d \in D \wedge d + t \in D\}. \quad (1)$$

The MTP of t in D is formed by the set of all points ind D that can be translated by t so that the translated points are also included in D. Negating the translation t produces the same MTP, that is, $MTP(t, D) = MTP(-t, D)$ [11], therefore only MTPs for translations that are greater than the zero vector are considered. All difference vectors between points in a point-set referred to in this paper are also assumed to be greater than the zero vector. All MTPs in a point-set can be computed with the SIA algorithm [11].

There are at most $n(n-1)/2$ MTPs in a point-set with n points. This occurs when differences between any pair of points are distinct. Conversely, there are at least $n - 1$ MTPs in a point-set. The minimum number of MTPs occurs in a point-set where all points are placed on a line with a constant distance between adjacent points. These extreme cases are unlikely to occur in point-sets representing music, however, they are useful for analysis of algorithms that compute MTPs. A point-set with a minimum number of MTPs is denoted by D_{min} and a point-set with a maximum number of MTPs is denoted by D_{max}.

A pattern P may have multiple occurrences in a point-set D. The set of all occurrences of P in D is represented by the *translational equivalence class* (TEC) [11] of the pattern. Two patterns A and B are considered translationally equivalent if, and only if, there exists a translation t such that $A = B + t$. Translational equivalence of A and B is denoted by $A \equiv_{\mathrm{T}} B$. The translational equivalence of patterns can be compared using their *vectorized representations* [11]:

$$VEC(P) = \langle p_2 - p_1, p_3 - p_2, \ldots, p_l - p_{l-1} \rangle, \quad (2)$$

where p_i denotes the ith point of a lexicographically sorted pattern P of length l. Two patterns are translationally equivalent if and only if their vectorized representations are equal [11]. The TEC of a pattern P in a point-set is the set of all subsets that are translationally equivalent to P:

$$TEC(P, D) = \{Q \mid Q \equiv_{\mathrm{T}} P \wedge Q \subseteq D\}. \quad (3)$$

The SIATEC algorithm can be used to compute the TECs of all MTPs in a point-set [11]. The TEC of a pattern P