A STRUCTURAL SIMILARITY INDEX BASED METHOD TO DETECT SYMBOLIC MONOPHONIC PATTERNS IN REAL-TIME

Nishal Silva and Luca Turchet

Department of Information Engineering and Computer Science
University of Trento, Italy
nishal.silva@unitn.it | luca.turchet@unitn.it

ABSTRACT

Automatic detection of musical patterns is an important task in the field of Music Information Retrieval due to its usage in multiple applications such as automatic music transcription, genre or instrument identification, music classification, and music recommendation. A significant sub-task in pattern detection is the *real-time* pattern detection in music due to its relevance in application domains such as the Internet of Musical Things. In this study, we present a method to identify the occurrence of known patterns in symbolic monophonic music streams in real-time. We introduce a matrix-based representation to denote musical notes using its pitch, pitch-bend, amplitude, and duration. We propose an algorithm based on an independent similarity index for each note attribute. We also introduce the Match Measure, which is a numerical value signifying the degree of the match between a pattern and a sequence of notes. We have tested the proposed algorithm against three datasets: a human recorded dataset, a synthetically designed dataset, and the JKUPDD dataset. Overall, a detection rate of 95% was achieved. The low computational load and minimal running time demonstrate the suitability of the method for real-world, real-time implementations on embedded systems.

1. INTRODUCTION

The presence of repeating patterns is one of the most significant aspects of music, as humans recognize structure in music mainly through the perception of repetition within a piece of music [1, 2]. One of the most studied topics in the field of Music Information Retrieval (MIR) is the detection of patterns [3, 2], and it has a multitude of applications such as computational musicology, audio fingerprinting and indexing, plagiarism detection, music classification, music recommendation, and music cognition. Although many researchers have presented algorithms capable of detecting repeated patterns in entire recordings or compositions, there seems to be a lack of attention devoted towards real-time implementations, i.e., when the analysis needs to be performed on the fly, at the exact moment in which the musical pattern is generated - usually by a musical instrument.

Real-time pattern detection is an integral part in Internet of Musical Things (IoMusT) applications [4] on the smart musical instruments [5]. These are an emerging class of digital musical instruments positioned at the intersection with Internet of Things devices, which are characterized by embedded systems dedicated to audio processing tasks (e.g., [6] and [7]). The intelligence embedded in such instruments could be used to extract musical patterns in real-time from the musician's output and immediately repurpose them into controls for other connected peripherals such as smoke machines, stage lights, visuals or smartphones of audience members in participatory live music contexts. To date, scarce research has been conducted to address these kinds of scenarios envisioned in IoMusT research, mainly due to the lack of appropriate real-time tools with constrained capabilities (such as limited computing power, limited memory, limited I/O ports, and requirements for minimum power consumption).

The algorithm presented by this paper is able to recognize monophonic musical patterns in a live musical environment utilizing digital musical instruments [5]. A modified version of the matrix-based representation method introduced in [8] is used to denote the musical notes. The proposed method draws inspiration from Computer Vision techniques and introduce a novel use for the Structural Similarity Index Measure (SSIM): a metric used to measure the similarity between two images in the domain of Computer Vision.

Human performers are not always perfect, and there may be subtle deviations present when a pattern is played. Musicians may also insert notes, or double some notes to suit their expressive need at the moment of playing. With the understanding that these nuances contribute to the expressiveness of the music, we have taken several steps to account for such deviations. We employ a dynamic window to extract sequences from the incoming music stream to account for any extra notes. We have also introduced a weighted summation and a threshold to obtain the final metric. The size of the dynamic window, threshold value, as well as the weights may be enforced as strictly, or as leniently as the user desires.

Several researchers have explored the use of structural similarity in the domain of music. The authors of [9] present a method employing structural similarity to summarize digital media by the frequency of occurrence of clusters within the media. The study reported in [10] presents a structural similarity measurement method between two recordings to enable audio-based analysis and retrieval. However, to our best knowledge, the usage of the structural similarity index to detect musical patterns in real-time has not been investigated yet.

2. RELATED WORK

There have been a multitude of research published on the detection of repeated patterns in music. However, most of these studies are aimed towards the detection of repeated patterns in recorded music.

The authors of [11] present a geometric approach to detect patterns in both monophonic and polyphonic music. The symbolic representations of music is used, and the method inspects all displacements between note pairs, i.e., if there is a pattern occurring twice in the piece, A and B - the distances from all notes in A to their counterparts in B should be the same. The chroma vector is used to cluster the repeated patterns. To tackle transposed patterns, the authors introduce a transposition of all chroma vectors to a single chroma vector.

The algorithm presented in [12] employs a self similarity matrix to identify repetitive musical patterns. This method is able to work with audio, or symbolic representations. A chromagram is obtained for the windowed signal, and the key-transposition invariant self-similarity matrix is computed. A scoring and threshold is performed followed by a grouping by occurrence.

The authors of [13] present a point-set comparison algorithm where the music is presented in the form of a multi dimensional point-set. The algorithm uses two-dimensional points, (t, p), where each point represents a single note or sequence of tied notes whose onset time is t in tatums and whose morphetic pitch is p. The authors also introduce the maximal translatable patterns vector, which is the set of points in the dataset that can be translated by a vector to give other points. This greedy compression algorithm is able to find the best maximal translatable patterns, append them to a new vector, and remove these points from the dataset.

Most existing literature are aimed towards the identification of repeating patterns present in entire compositions and they operate with knowledge of the composition in its entirety. However, we are aiming towards a system to be used in a live environment, with no prior knowledge of the incoming notes beforehand. Due to this reason, a direct comparison cannot be computed between the proposed method and the state of the art.

In our earlier study, we presented a comparison between a deterministic method and several machine learning methods designed to detect patterns in symbolic real-time music [8]. The deterministic method performed a simple boundary check. We also evaluated different machine learning approaches; which include a single neural network trained to detect all patterns, multiple neural networks trained to detect each pattern, a convolutional neural network, and a recurrent neural network. The deterministic system and the recurrent neural network showed the best results.

Through this work, we aim to overcome some limitations encountered in our previous study. The reduction of false detections, and the ability to detect transposed patterns are some of the key improvements of the proposed method. We also wish to reduce the set-up time when compared with machine learning methods.

3. REPRESENTATION OF MUSICAL NOTES

We use a modified version of the matrix-based representation introduced in our earlier study [8], which is inspired by the MIDI protocol and uses three attributes to denote a single musical note - namely the *Pitch, Amplitude*, and *Duration*. In the present study, we introduce an additional attribute: *Pitch Bend* - used to denote the graceful increase or decrease of the pitch of a musical note. A column matrix is used to encapsulate the four attributes and denote a single musical note. The four attributes can be described as follows:

- **Pitch:** The pitch of a note is a perceptual quality which allows for it to be deemed *higher* or *lower* than other notes, and enables a note to be ordered on a frequency-related scale [14, 15]. We use the 12 tone equal temperament (12-TET) tuning system, and the pitch is obtained from the MIDI number.

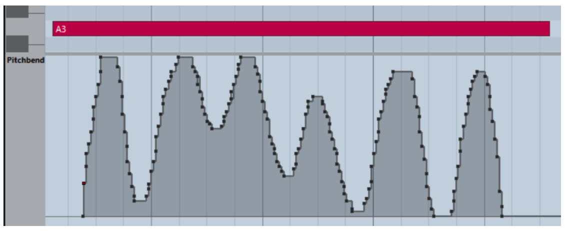

Figure 1: Pitch bend values of an example musical note

- **Pitch-bend:** Pitch-bend is a musical technique which is used to obtain vocal-like expressive characteristics and articulations from musical instruments by increasing or decreasing the pitch of a note. String instruments achieve this effect by bending the desired string, and keyboard style instruments achieve this effect by the use of a pitch controller. Pitch-bend is similar to the music concept of *glissando*.

  To denote the pitch bend, we use the MIDI pitchwheel parameter. Our algorithm operates under the assumption that the common interval of ± 2 semitones for the pitchwheel range will be used [16].

  To denote a smooth and graceful pitch bend, MIDI instruments encode the pitch bends as a series of MIDI messages with fine increments or decrements. Using all pitchwheel values is not feasible as it would render extremely long sequences with arbitrary lengths. To overcome this issue, we have decided to only use the local minima or maxima to denote the pitch bend in order to retain its shape with a minimal amount of data. Fig. 1 shows an illustration of a pitch bend done on a single musical note. For this particular note, the pitch bend changes are represented by 158 MIDI messages. We use the values corresponding to the 6 peaks and 5 troughs instead of all 158 values. According to our representation standard, the Pitch bend values for Fig. 1 are [*0, 8191, 800, 8191, 4454, 8191, 2114, 6096, 266, 7451, 61, 7492, 0*].

- **Amplitude:** The amplitude corresponds to the loudness of the note. We consider the relative loudness in each note of a sequence as an important attribute in determining the match with a pattern. MIDI velocity is used to represent the amplitude.

  The velocity value, as defined in the MIDI standard, corresponds to the rate at which a keyboard key is pressed [16, 17]. This action on an acoustic instrument will render a louder, or a softer sound. Although MIDI velocity may be used to control multiple parameters in a synthesizer [17], we operate under the assumption that MIDI velocity is used to control only the amplitude.

  In music, dynamics refer to the variation in loudness between notes and phrases, and are represented using dynamic markers, which may range from *pianississimo*, which refers to *very very quiet* playing, to *fortississimo*, which refers to *very very loud* playing [18]. Table 1 shows a complete list of dynamics markers, along with their symbols and corresponding MIDI velocity values as mapped by the popular music production software Logic Pro [19].

- **Duration:** The duration of a note is the time duration where the note undergoes its gradual decay. For our computations, we represent the duration in seconds.

Table 1: *MIDI velocity and dynamics mapping in Logic Pro 9 (retrieved from [19]).*

Name	Level	Marker	Velocity
fortississimo	very very loud	*fff*	127
fortissimo	very loud	*ff*	112
forte	loud	*f*	96
mezzo-forte	average	*mf*	80
mezzo-piano	average	*mp*	64
piano	quiet	*p*	48
pianissimo	very quiet	*pp*	32
pianississimo	very very quiet	*ppp*	16

Eq. (1) provides an example of a single musical note denoted using the representation mechanism introduced above. For any i^{th} note in a sequence N_i, the pitch is denoted by P_i, the current value of pitch-bend is denoted by B_i, the amplitude is denoted by A_i, and the duration is denoted by D_i.

$$N_i = \begin{bmatrix} P_i \\ B_i \\ A_i \\ D_i \end{bmatrix}. \tag{1}$$

Accordingly, eq. (2) shows a sequence of notes N, which are represented using the defined format:

$$N = \begin{bmatrix} \begin{matrix} P_i \\ B_i \\ A_i \\ D_i \end{matrix}, \begin{matrix} P_{i+1} \\ B_{i+1} \\ A_{i+1} \\ D_{i+1} \end{matrix}, \begin{matrix} P_{i+2} \\ B_{i+2} \\ A_{i+2} \\ D_{i+2} \end{matrix}, \dots \end{bmatrix} \tag{2}$$

It should be noted that the MIDI note number 0 is used to represent the C_{-1} note, which is an octave below the $C_0 = 16.35Hz$ note [20]; just below the lower limit of human hearing, and is well outside the frequency range of most contemporary musical instruments. Therefore, we can safely rule out the chances of a C_{-1} note occurring in contemporary music, and use its MIDI number to represent a pause.

4. MUSICAL PATTERNS

We define a repeating musical pattern to be an ordered sequence of notes and pauses which occur at least twice in a piece of music. The repetitions of patterns will be shifted in time, and may be transposed. However, in a real life performance, musicians may repeat notes or insert extra notes to better express the emotions associated with the music piece. Such cases may also be perceived as a pattern by the listener for as long as the melody and the emotion of the music is preserved.

Let us consider the first four notes of the C major scale as shown in Figure 2. If we assume that all notes have an equal moderately loud amplitude (refer Table 1), no pitch bends, and a tempo of ♩ $= 120\ bpm$, we can denote S in the matrix-based representation as shown below;

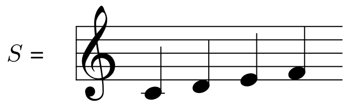

Figure 2: First four notes of the C major scale

$$S = \begin{bmatrix} 60 & 62 & 64 & 65 \\ 0 & 0 & 0 & 0 \\ 80 & 80 & 80 & 80 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}. \tag{3}$$

For our algorithm, several conditions must be met for a sequence of notes to be a match with pattern S:

- The sequence and the pattern must be of identical length, or the sequence may only have less than the allowed number of notes added or doubled.
- The final metric, which is a weighted summation, should be higher than the defined hard threshold.

5. THE STRUCTURAL SIMILARITY INDEX

The Structural Similarity Index Measure (SSIM) is a Computer Vision technique to measure the *similarity* between two images. The SSIM provides advantages over standard comparison methods such as a Mean Squared Error, and it derives inspiration from the human visual perception to identify structural information from a scene [21]. The SSIM index for two image signals x and y, is presented as follows: [21];

$$SSIM_{(x,y)} = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{({\mu_x}^2 + {\mu_y}^2 + C_1)({\sigma_x}^2 + {\sigma_y}^2 + C_2)}. \tag{4}$$

C_1 and C_2 are constants to ensure stability when the denominator reaches 0 for each measure. μ_x, and μ_y is measured by averaging over all the pixel values. ${\sigma_x}^2$ and ${\sigma_y}^2$ is measured by computing the variance of all the pixel values, and the covariance between x and y are denoted by σ_{xy}, as shown in eq. (6). They are defined for each pixel i of an arbitrary variable x as below.

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{5}$$

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}. \tag{6}$$

The SSIM outputs a single value between -1 and $+1$. A value of $+1$ indicates that the given images are identical, and a value of -1 indicates that the given images are very different.

6. BICUBIC INTERPOLATION

Interpolation is a common technique used in Computer Vision to resize images. There are several interpolation methods available for image resizing, and bicubic interpolation is a common technique used by several popular image processing software [22].

In the context of computer vision, bicubic interpolation calculates a weighted average of the surrounding 4x4 pixel square to

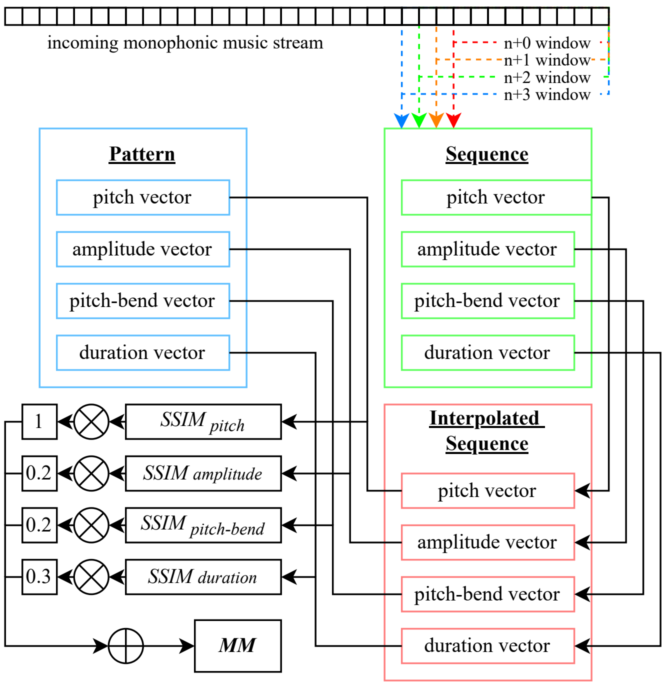

Figure 3: Flow chart of the proposed algorithm.

calculate the interpolated value for an unknown pixel u. Each surrounding pixel is assigned a weight based on its distance to the destination pixel [23]. The kernel for the bicubic interpolation is illustrated in eq. 7, where d is the distance between the interpolated point and the grid point [22, 23].

$$u(d) = \begin{cases} 3/2|d|3 - 5/2|d|2 + 1, & 0 \leq |d| < 1, \\ -1/2|d|3 - 5/2|d|2 - 4|d| + 2, & 1 \leq |d| < 2, \\ 0 & 2 < |d|. \end{cases} \quad (7)$$

7. PROPOSED METHOD

We propose an algorithm to detect if a sequence of notes is a match to a given pattern according to the conditions discussed in Section 4. The algorithm obtains an individual SSIM value for each set of attributes belonging to the sequence and the pattern respectively. To account for extra notes or removed notes, we have utilized a dynamic window to obtain sequences from the incoming music stream.

7.1. Dynamic Window

Musicians may add extra notes or remove some notes from a pattern based on their expressive needs at a live performance. With the understanding that a sequence can still be perceived as a pattern despite some extra notes being present, we use a dynamic window to obtain sequences to compute the SSIM against each pattern.

The size of the window can be used as a control parameter to allow a p number of notes to be added to the pattern. In our simulations, we used the value of $0 \leq p \leq 3$, which allows for a maximum of 3 extra notes to be added. As illustrated in Fig. 3, if the length of a pattern in n, multiple sequences will be extracted from the music stream, each with lengths $n+0$, $n+1$, $n+2$, and $n+3$. The value of p for our experiments was chosen arbitrarily, and it may be adjusted to make the system as strict, or as lenient as the user desires.

7.2. Resizing using Bicubic Interpolation

One of the primary criterion for the SSIM is that the two matrices being compared has to be identical in size. As the windowed segments may vary in length by a factor of p, we perform a bicubic interpolation to resize the sequences of length $n+p$, to the length of the pattern, which is n.

In our experimental setup, four windowed sequences are extracted from the incoming music stream for each pattern as $0 \leq p \leq 3$. Interpolation is not necessary for the $p = 0$ sequence. For all other sequences where $p = 1, 2, 3$, the interpolation is performed as explained in eq. 7. Following which, the interpolated vectors are used to compute the SSIM values with the pattern attribute vectors as illustrated in Fig. 3.

7.3. SSIM Computation

The interpolated sequence and the pattern is first split into pitch, amplitude, pitch-bend, and duration vectors. This step is taken to ensure that each attributes contribute independently to the final measure. We then compute the SSIM between each pair of the pitch, amplitude, pitch-bend and duration vectors of the interpolated sequence and each pattern, as illustrated by Fig. 3.

For each vector pair x and y, we calculate μ_x, μ_y, σ_x, and σ_y as presented in eq. 4. We then use the calculated μ_x, μ_y, σ_x, and σ_y values to compute the SSIM value for x and y, for all four vector pairs.

The constants C_1, and C_2 are small constants, included to avoid instability when ${\mu_x}^2 + {\mu_y}^2$, and ${\sigma_x}^2 + {\sigma_y}^2$, respectively, is very close to zero. For our evaluations, we used the arbitrary values $K_1 = 0.01$, and $K_2 = 0.03$, the same values used in the original publication. Furthermore, we selected the value $L = 127$, which is the dynamic range for pitch and amplitude. C_1 and C_2 are defined as [21]:

$$C_1 = (K_1 L)^2, \quad (8)$$

$$C_2 = (K_2 L)^2. \quad (9)$$

As discussed above, we obtain four SSIM values for each pair of note attributes. For each sequence x, and pattern y, the SSIM between the two pitch vectors is $SS_{p(x,y)}$, the SSIM between the two amplitude vectors is $SS_{a(x,y)}$, the SSIM between the two pitch-bend vectors is $SS_{pb(x,y)}$, and the SSIM between the two duration vectors is $SS_{d(x,y)}$. We then use the four SSIM values to compute the Match Measure (MM) for between each sequence and pattern.

7.4. Match Measure

MM is is obtained through a weighted summation of all four SSIM values as illustrated in eq. 10. This step ensures that all attributes in the sequence and the pattern make a contribution based on their importance to the definition of a pattern. A greater difference in one set of attributes will lead to a smaller MM. If all attributes are

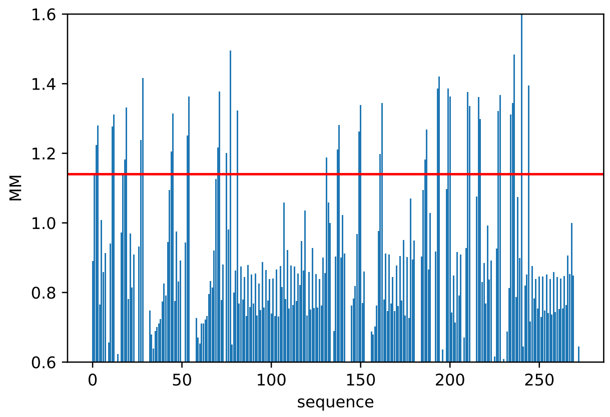

Figure 4: MM values for each windowed sequence for a composition in the human dataset.

identical, a value of $MM = 1.7$ is obtained, and for every deviation from the pattern, the MM will decrease. The lowest possible value is $MM = -1.7$

$$MM_{(x,y)} = SS_{p(x,y)} + 0.2 \times SS_{a(x,y)} + \\ 0.2 \times SS_{pb(x,y)} + 0.3 \times SS_{d(x,y)}. \tag{10}$$

Each note attribute contribute to the MM independently. As a result, any deviation in one attribute will lower the MM value, which allows us to identify the similarity between a note sequence and a pattern.

7.5. Tolerances and Weights

As discussed above, MM is obtained through a weighted summation of the SSIM values of the four note attribute vector pairs. The weights for each SSIM value, as shown by eq. 10, were selected arbitrarily based on the relative contribution each attribute makes to the definition of the pattern.

The on-time pitch pairs are the most important attribute to determine if a sequence and a pattern is a match [24]. We identified empirically that the amplitude contributes less to the definition of a pattern. It should be noted that the user may change these weights based on the importance they give to each note attribute.

Fig.4 shows the MM values for a single pattern for a composition in the dataset. It is clear that some sequences have a significantly higher MM values, and we defined the final threshold values based on the relative difference of MM values. Each composition requires a unique threshold value to determine the allowable deviations from a pattern. The users may set the threshold value to be as strict, or as lenient as they desire.

To illustrate on the algorithm let us consider an example from the JKUPDD dataset, specifically Beethoven's Piano Sonata in F minor, and consider one repeating pattern, 29 notes in length. Let us name the pattern pitch vector as P_p, the amplitude vector as P_a, the pitch-bend vector as P_{pb}, and the duration vector as P_d:

$$P_p = [69,\ 69,\ 67,\ 65,\ 62,\ 64,\ 65,\ 67,\ 69,\ 62,\ 64,\ 65, \\ 67,\ 69,\ 65,\ 67,\ 67,\ 64,\ 65,\ 67,\ 67,\ 69,\ 65,\ 67, \\ 65,\ 64,\ 65,\ 64,\ 65] \tag{11}$$

$$P_a = [90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90, \\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90, \\ 90,\ 90,\ 90,\ 90,\ 90] \tag{12}$$

$$P_{pb} = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0, \\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0] \tag{13}$$

$$P_d = [0.72,\ 0.24,\ 0.48,\ 0.48,\ 0.24,\ 0.24,\ 0.24,\ 0.24, \\ 0.48,\ 0.72,\ 0.24,\ 0.24,\ 0.24,\ 0.24,\ 0.24,\ 1.44, \\ 0.48,\ 0.48,\ 0.48,\ 0.72,\ 0.24,\ 0.48,\ 0.48,\ 0.72, \\ 0.24,\ 0.48,\ 0.96,\ 0.48,\ 0.48] \tag{14}$$

Within the piece, there is a repetition of the pattern where two doubled notes are present, making the sequence 31 notes long. Let us illustrate this sequence pitch vector as S_p, the amplitude vector as S_a, the pitch-bend vector as S_{pb}, and the duration vector as S_d:

$$S_p = [65,\ 69,\ 69,\ 67,\ 65,\ 62,\ 64,\ 65,\ 67,\ 69,\ 62,\ 64, \\ 65,\ 67,\ 69,\ 65,\ 67,\ 67,\ 64,\ 65,\ 67,\ 67,\ 69,\ 65, \\ 67,\ 65,\ 64,\ 65,\ 64,\ 65,\ 69] \tag{15}$$

$$S_a = [90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90, \\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90, \\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90] \tag{16}$$

$$S_{pb} = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0, \\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0] \tag{17}$$

$$S_d = [0.48,\ 0.72,\ 0.24,\ 0.48,\ 0.48,\ 0.24,\ 0.24,\ 0.24, \\ 0.24,\ 0.48,\ 0.72,\ 0.24,\ 0.24,\ 0.24,\ 0.24,\ 0.24, \\ 1.44,\ 0.48,\ 0.48,\ 0.48,\ 0.72,\ 0.24,\ 0.48,\ 0.48, \\ 0.72,\ 0.24,\ 0.48,\ 0.96,\ 0.48,\ 0.48,\ 0.72] \tag{18}$$

The interpolation resizes the pitch, amplitude, pitch-bend and duration vectors of the sequence to fit the length of the corresponding pattern vectors. The interpolated sequence has a pitch vector I_p, an amplitude vector I_a, a pitch-bend vector I_{pb}, and a duration vector I_d as follows:

$$I_p = [65,\ 69,\ 68,\ 66,\ 64,\ 62,\ 64,\ 66,\ 68,\ 64,\ 63,\ 64, \\ 66,\ 68,\ 65,\ 67,\ 66,\ 64,\ 65,\ 67,\ 67,\ 67,\ 66,\ 65, \\ 64,\ 64,\ 64,\ 64,\ 68] \tag{19}$$

$$I_a = [90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90,\ 89, \\ 90,\ 90,\ 90,\ 90,\ 90,\ 89,\ 90,\ 90,\ 90,\ 90,\ 90,\ 90, \\ 90,\ 90,\ 90,\ 90,\ 90] \tag{20}$$

$$I_{pb} = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0, \\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0] \tag{21}$$

$$I_d = [0.50,\ 0.66,\ 0.29,\ 0.48,\ 0.40,\ 0.24,\ 0.24,\ 0.24, \\ 0.38,\ 0.63,\ 0.38,\ 0.24,\ 0.24,\ 0.24,\ 0.28,\ 1.35, \\ 0.48,\ 0.48,\ 0.55,\ 0.55,\ 0.34,\ 0.48,\ 0.61,\ 0.43, \\ 0.4,\ 0.84,\ 0.57,\ 0.48,\ 0.70] \tag{22}$$

The interpolated vectors, and the pattern vectors are of identical length, and the SSIM between each vector pair can now be calculated. The obtained SSIM values for the above pattern and sequence are:

- SSIM between P_p (eq:11) and I_p (eq. 19) = 0.98175
- SSIM between P_a (eq:12) and I_a (eq. 20) = 0.99998
- SSIM between P_{pb} (eq:13) and I_{pb} (eq. 21) = 1.0
- SSIM between P_d (eq:14) and I_d (eq. 22) = 0.32245

Following the computation of the SSIM values, we can compute the MM between the pattern and the sequence as shown by eq. 10. The final metric is $MM = 1.478481$.

8. DATASET

We used three separate datasets to evaluate the system; namely the Human Dataset, the Synthesized Dataset, and the Johannes Kepler University Patterns Development Database (JKUPDD). Each dataset is able to represent different application domains and have their own unique characteristics which are discussed below.

8.1. Human Dataset

This dataset was recorded by one of the authors and it contains subtleties in note attributes that can only be achieved with human playing. The dataset contains staccato and legato styled playing, changes in amplitude, as well as pitch-bends. The Human Dataset consists of 45 tracks. Each track was recorded so that they contain one, or several repeating patterns with other sequences in between. There are 15 distinct patterns with a total of 300 repetitions which contain extra notes, doubled notes as well as added pitch bends. The pattern occurrences were annotated at the time of recording. The dataset was recorded using an M-Audio Oxygen Pro Mini MIDI keyboard, with Cubase LE AI Elements 8.

8.2. Synthesized dataset

We constructed an ad-hoc dataset from 16 symbolic monophonic tracks of contemporary songs and music pieces to evaluate the system. The dataset is an extension of the dataset used in our earlier research [8], and it spans across multiple styles of music including pop, rock, heavy metal, instrumental songs, classical pieces, and folk songs. We identified repeating patterns in each track that would warrant the use of the proposed system. There are 36 distinct patterns present with a total of 753 repetitions. All pattern occurrences are either transposed, or identical to the ground truth. Pattern repetitions were identified by listening, and annotated manually. The symbolic data was obtained through the community contributed, online resource *Ultimate Guitar*[1].

8.3. JKUPDD dataset

The Johannes Kepler University Patterns Development Database is an industry standard, annotated, open source dataset used in many pattern recognition tasks[2]. It contains 5 classical pieces in both monophonic, and polyphonic versions with repeating patterns, and has been widely used to evaluate algorithms. The ground truth, as well as all pattern occurrences are provided in the dataset. We used the monophonic MIDI version of the dataset to evaluate our system.

We believe that the three datasets are able to provide a good base to evaluate the proposed method due to their unique characteristics. The datasets are able to simulate situations that may arise in a live performance or in programmed music. We have made all datasets and source codes publicly available[3]. Figure 5 is an example of a repeating patterns present in the dataset, which contains long sequences of notes with varying durations.

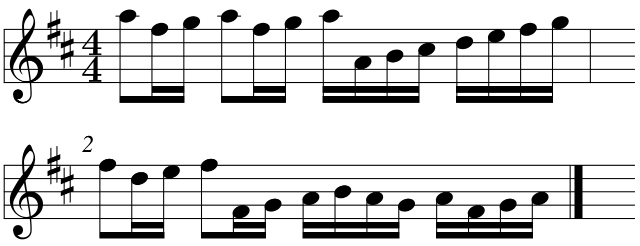

Figure 5: A segment from Pachelbel's Canon in D: a test pattern used in the synthetic dataset

9. RESULTS

In order to simulate a real-time operation, we streamed each track to the system, note by note. Note sequences were obtained, interpolated, and MM was computed on the fly. The results of our simulations are presented in Table 3, along with other details on the datasets - such as the total number of patterns, number of distinct patterns, total length of patterns and non-patterns as well as the ratio between patterns/ non-patterns and tracks in Table 2. Results show that the proposed system is capable of identifying a majority of patterns in the Human and JKUPDD datasets, and 100% of patterns in the Synthetic Dataset.

We introduced a comparison between several probabilistic and deterministic methods for the same task in a previous publication [8]. The highest performing methods of said study, namely the Deterministic System (Det), and the recurrent Neural Network based System (RNN) were used as benchmarks to assess the accuracy of the proposed system.

Det consists of a series of boundary checks for each note attribute and sequence and a pattern are deemed as a match if all note attributes are within the specified tolerances from each other. RNN consists of a recurrent neural network with an input layer whose length is equal to the longest pattern, two LSTM layers, a dense layer, a dropout layer, and a dense output layer with a softmax activation function. As deep learning requires large training sets for maximum accuracy, a large training dataset was constructed by artificially introducing variations for the selected patterns. We also generated sequences which are non patterns and labelled them accordingly.

Table 3 provides a comparison of the precision, recall and F-measure of each method globally, and for each dataset. Results show that the proposed system was able to identify an excellent percentage of patterns in the dataset. Det and RNN both suffered due to their inability to detect transposed patterns. Moreover, RNN showed a significant amount of false positives despite its good performance in recognizing patterns.

As illustrated by Table 4, the proposed system showed a low computational load when compared with compared with Det and RNN. All tests were performed on a workstation laptop with an Intel core $i7-11800H$ $(2.3GHz)$ processor with $16GB$ of memory.

[1]`http://ultimate-guitar.com`
[2]`https://github.com/ns2max/jkupdd_dataset`
[3]`https://github.com/ns2max/ssimpatt`

Table 2: Overview of the utilized datasets.

	Human recorded dataset	**Synthesized dataset**	**JKUPDD**
Number of tracks	45	16	5
Number of distinct patterns	15	36	31
Total number of patterns	300	753	137
Total length of all patterns	296	644	1430
Total length of all tracks	9872	16872	3547
Distribution length of patterns and tracks	0.029	0.038	0.403
Distribution length of non-patterns and tracks	0.971	0.962	0.597

However, these results show that the proposed method is suitable for a real-time application on a resource constrained embedded system as well [6].

Table 3: *Comparison of the performances between MM, Det, and RNN for each dataset.*

		MM	**Det**	**RNN**
Combined Dataset	Accuracy	95.0%	53.6%	51%
	Precision	0.95	0.94	0.86
	Recall	0.97	0.53	0.51
	F-measure	0.96	0.68	0.64
Human Dataset	Accuracy	96.6%	56%	55%
	Precision	0.93	0.94	0.82
	Recall	0.98	0.54	0.54
	F-measure	0.96	0.69	0.65
Synthetic Dataset	Accuracy	100%	51.8%	49.4%
	Precision	0.97	0.95	0.89
	Recall	0.98	0.51	0.49
	F-measure	0.97	0.66	0.63
JKUPDD	Accuracy	70.5%	57.3%	50.7%
	Precision	0.95	0.90	0.76
	Recall	0.97	0.61	0.59
	F-measure	0.96	0.73	0.66

Table 4: *Comparison between average running times and average memory usage of different pattern detection methods.*

	MM	**Det**	**RNN**
Average running time (in ms)	0.6	2.1	24.8
Average memory usage (in MB)	69	62	384

10. DISCUSSION AND CONCLUSION

In this paper, we have presented an algorithm capable of identifying the presence of predefined patterns in monophonic music in real-time. We employ a dynamic window to allow a musician to double notes, or insert extra notes to a pattern. The notes are represented using the four attributes: pitch, amplitude, pitch-bend, and duration. The SSIM is calculated independently for each attribute pair of the pattern and the interpolated sequence. Finally, a weighted summation is performed to obtain the final metric MM.

The individual SSIM calculation allows for each note attribute pair to contribute independently to the final metric. We have assigned weights for each SSIM measure based on their relative importance to the definition of a pattern. The proposed system was capable of identifying approximately 95% of patterns in the combined dataset.

The algorithm was able to successfully detect approximately 96% of the patterns in the Human dataset. This dataset was recorded by human performers, and it contains many subtle variations that cannot be synthetically achieved. Due to the dynamic window, the algorithm was able to detect instances where extra notes were added to pattern repetitions.

The proposed method was able to successfully identify the occurrence of approximately 70% of the patterns in the JKUPDD dataset. The JKUPDD dataset contains repetitions of patterns, transpositions, as well as variations of patterns.

There are many instances where repetitions of patterns contain doubled notes as well as extra notes and pauses. The proposed method was able to identify such patterns due to its usage of the dynamic window. However, the algorithm failed to identify pattern repetitions which had notes removed. The JKUPD dataset also contains pattern repetitions whose durations are highly altered. These patterns generated a lower MM than the defined threshold and were note detected.

The synthetic dataset had a 100% detection rate, as variations such as extra notes and pauses are absent. Although we introduced deviations for note attributes in the dataset, we did not introduce any pauses, doubled notes, or changes in pitch to the synthetic dataset to accurately represent music that may be programmed or computer generated. The proposed system was able to identify all occurrences of patterns and their transpositions in the synthetic dataset.

It is evident from Table 2 that the distribution ratio between patterns and tracks is low when compared with that between non-patterns and tracks. These figures show that non-pattern sequences occupy a majority of the dataset, much like a real world performance.

Table 3 shows the performance for each method for the global dataset as well as for each individual datasets. The proposed system showed a better accuracy when compared with Det and RNN in all fronts as RNN and Det were unable to detect transposed versions of patterns. It is possible to configure Det to identify transposed patterns by utilizing the pitch differences instead of the absolute pitch. However, we aim to develop a system that is able to identify the degree of match between a sequence and a pattern, and Det is unable to accomplish this due to its nature. RNN showed a good percentage of detections, but a relatively high number of false positive detections was also present. This, coupled with the fact that an extremely large dataset is required to adequately train a machine learning model, as well as the high training time, are drawbacks to any machine learning based model in this context. Such limitations inhibit these systems from a real-world, real-time implementation.

This research intends to improve several limitations that our previous study encountered [8]. The proposed algorithm is able to detect patterns and establish a degree of match between a sequence and a pattern while having a better accuracy and lower computational cost over our previous work. The average computational time is well within the 10 millisecond limit for accepted latency in real-time digital musical instruments [7]. The proposed system also shows advantages as there is minimal set up when compared to Neural Network-based methods which usually require a considerable time and data for training.

However, the proposed method contains several limitations that we wish to overcome to allow more artistic freedom to a musician. We hope to further increase the accuracy as well as reduce the number of false detections. The current version of the algorithm requires an input from the user to set the threshold value for each composition. We hope to improve the system to dynamically adjust its weights, as well as the threshold values for an improved and completely automated detection of patterns.

For future work, we plan to create a database of patterns that is more suited to evaluate pattern recognition tasks by interviewing different musicians from various musical and ethnic backgrounds, influences, playing styles, and age groups. As the playing styles and subtle variations will be unique to each individual musician, this dataset will be able to accurately represent most variations we might encounter in a real life performance.

We also plan to implement an intelligent instrument capable of detecting predefined patterns in real-time, where the proposed algorithm will run on an embedded system working in tandem with a MIDI keyboard controller. Upon identifying a successful match, a control message will be sent to the predefined peripherals. The MIDI data will then be passed through to other plugins or synthesizers to produce sound.

11. REFERENCES

[1] R. Dannenberg and N. Hu, "Linear time for discovering non-trivial repeating patterns in music databases," in *Proceedings of the Third International Conference on Music Information Retrieval*, 2002, pp. 63–70.

[2] I. R. Ren, H. V. Koops, A. Volk, and W. Swierstra, "In search of the consensus among musical pattern discovery algorithms," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017, pp. 23,27.

[3] J. A. Burgoyne, I. Fujinaga, and J. S. Downie, *Music Information Retrieval*, pp. 213–228, Wiley-Blackwell, 2015.

[4] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet, "Internet of Musical Things: Vision and Challenges," *IEEE Access*, vol. 6, pp. 61994–62017, 2018.

[5] L. Turchet, "Smart Musical Instruments: vision, design principles, and future directions," *IEEE Access*, vol. 7, pp. 8944–8963, 2019.

[6] L. Turchet and C. Fischione, "Elk Audio OS: an open source operating system for the Internet of Musical Things," *ACM Transactions on the Internet of Things*, vol. 2, no. 2, pp. 1–18, 2021.

[7] A. Mcpherson, R. Jack, and G. Moro, "Action-sound latency: Are our tools fast enough?," in *Proceedings of the Conference on New Interfaces for Musical Expression (NIME)*, 2016, pp. 20–25.

[8] N. Silva, C. Fischione, and L. Turchet, "Towards real-time detection of symbolic musical patterns: Probabilistic vs. deterministic methods," in *2020 27th Conference of Open Innovations Association (FRUCT)*, 2020, pp. 238–246.

[9] M. Cooper and J. Foote, "Summarizing popular music via structural similarity analysis," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003, pp. 127–130.

[10] J. P. Bello, "Measuring structural similarity in music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2013–2025, 2011.

[11] T. P. Chen and L. Su, "Discovery of repeated themes and sections with pattern clustering," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[12] O. Nieto and M. M. Farbood, "Music segmentation techniques and greedy path finder algorithm to discover musical patterns," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[13] D. Meredith, "Using siateccompress to discover repeated themes and sections in polyphonic music," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[14] C. J. Plack, A. J. Oxenham, and R. R. Fay, *Pitch - Neural Coding and Perception*, Springer Verlag, New York, 1 edition, 2005.

[15] A. Klapuri and M. Davy, *Signal Processing Methods for Music Transcription*, Springer Verlag, New York, 1 edition, 2006.

[16] "Official MIDI Specifications: General MIDI I," Available at https://www.midi.org/specifications/midi1-specifications/general-midi-specifications/general-midi-1.

[17] R. Dannenberg, "The interpretation of midi velocity," in *Proceedings of the 2006 International Computer Music Conference, San Francisco, CA*, 2006, pp. 193–196.

[18] G. Read, *Music Notation: A Manual of Modern Practice*, Allyn and Bacon, Boston, Massachusetts, 1969.

[19] "Logic Pro 9: User Manual," Available at https://help.apple.com/logicpro/mac/9.1.6/en/logicpro/usermanual/Logic%20Pro%209%20User%20Manual%20(en).pdf.

[20] "Official MIDI Specifications: MIDI Tuning," Available at https://www.midi.org/specifications/midi1-specifications/midi-1-addenda/midi-tuning-updated.

[21] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[22] A. Prajapati, S. Naik, and S. Mehta, "Evaluation of Different Image Interpolation Algorithms," *International Journal of Computer Applications*, vol. 58, no. 12, pp. 6–12, 10 2012.

[23] V. Patel and K. Mistree, "A Review on Different Image Interpolation Techniques for Image Enhancement," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 12, pp. 129–133, 12 2013.

[24] O. Lartillot and P. Toiviainen, "Motivic matching strategies for automated pattern extraction," *Musicae Scientiae*, vol. 11, no. 1_suppl, pp. 281–314, 2007.