Classical Music Analysis using AATMoF: An Automatic Algorithmic Transformed Motif Finder

Victor Robila[a]*, Stefan Nita[b], Joshua Cho[b], and Bogdan Nita[c]

[a]*Hunter College High School, New York City, USA;*

[b]*Kinnelon High School, Kinnelon, USA;*

[c]*Department of Mathematics, Montclair State University, Montclair, NJ, USA*

*victor.robila@gmail.com

Provide short biographical notes on all contributors here if the journal requires them.

Classical Music Analysis using AATMoF: An Automatic Algorithmic Transformed Motif Finder

Music is inherently related to mathematics and mathematical transformations of musical motifs have been found in many composers' works. Copyright law and musical analysis makes it beneficial to create a low-resource algorithm to automatically discover transformed motifs in musical pieces. We develop Automatic Algorithmic Transformed Motif Finder (AATMoF) and test it on Bach musical pieces. Results show that AATMoF based analysis either outperformed or was matched the accuracy of human analysis and was several orders of magnitude faster.

Keywords: motifs, musical analysis, copyright law, mathematical transformations

Subject classification codes: include these here if the journal requires them

1. Introduction

A musical motif is defined as the "smallest structural unit possessing thematic identity" [1]. Through their motifs' harmonic, melodic or rhythmic aspects, music pieces achieve their character. Most musical compositions use at least one motif while more complex pieces can have multiple ones with different lengths. Moreover, a motif can be slightly altered leading to further musical diversity .

Finding motifs in music is important for musical analysis and has applications in copyright law and musical composition [2]–[4]. In terms of copyright law, while the main body of a piece may be original, using a motif as the baseline of the piece can constitute as musical plagiarism. Finding motifs can also be very difficult and time-consuming when done manually even if the motif is already well defined. In addition, human analysis of finding motifs in a piece can be difficult for more complex pieces and could lead to mistakes. It is therefore important to develop methods of finding motifs automatically.

A musical motif can also be transformed geometrically within a composition: it can be repeated (translated horizontally), transposed (translated vertically), inverted (horizontally mirrored), in retrograde (vertically mirrored), in retrograde inversion (rotated by 180°) and even show up in these combinations but with different note values (dilated or contracted). A rigorous definition of these mathematical transformations can be found in section 3.3. Geometrically transformed motifs are especially interesting due to their unique structures and the fact that they could retain a musical piece's general melody while evading copyright detection efforts. Identifying the motifs and their transformations can also be helpful to composers attempting to replicate a certain style by creating original motifs and melodies on the transformational structure of another composer [5].

In this paper, we develop AATMoF, an Automatic Algorithmic Transformed Motif Finder, to analyze musical pieces in order to identify pre-defined motifs and their geometrical transformations. This program takes in the Musical Instrument Digital Interface (MIDI) data for any song, and the position of a selected motif, and automatically finds the location, inside of the song, of any occurrence of a geometrically transformed version of that motif. It also outputs a metric to quantify the similarity between all consecutive phrases with the same length as the motif in the inputted musical piece and the possible transformed instances of the motif. This metric ranges from 0, for an identical motif, to 25, for a musical phrase that share no notes in common with the motif. We test AATMoF on two pieces: Bach's Contrapunctus 7, and Prelude in C, and compare its performance to a human analysis of both pieces. Bach is an especially prolific user of motifs and often has complex structures within his music [6], [7]. The two pieces were selected due to their differences and contrasting complexity in the use of geometrical transformations of the main motif. This paper is part of a larger project that focuses on developing Bach-like

compositions using mathematically transformations of original motifs following the same geometrical structure utilized by Bach.

2. Previous Work

There have been several attempts at automatically identifying motifs in music. The Constrained String Mining Algorithm (CSMA) described in [10] works by taking in inputs of maximum and minimum motif length, maximum gap lengths between motifs, and minimum frequency thresholds to find motifs throughout the piece. This approach worked well for finding motifs for pieces in which they are not well defined and can be used in conjunction with our approach which looks for transformed instances of a defined motif. SSMiner (Similar Sequence Miner) is another approach that used a TreeMiner-like algorithm to identify transformed motifs [11]. Other methods such as the one described in [12] used MIDI files that analyzed chronologically. While these methods were successful in identifying short patterns, they become computationally infeasible as the patterns' length increases. Several other approaches have been tried in [13]–[15]. A full background on this field can be found in [16].

Octave	Note numbers											
	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

Figure 1: MIDI Note Conversion Chart [30]

Machine learning has also been used for music analysis and generation. Pituk explored motif extraction for melodic motifs and percussion sections using a Convolutional Neural Network for OP-Z data [17]. Liang and colleagues used a Long Short-Term Memory (LSTM)machine

learning model to produce Bach-like pieces [18]. We used their song format in our algorithm since it allowed us to have a simple way to find the difference between two musical phrases by subtracting the arrays created in the song format.

Motif extraction from data strings has been used in other fields, as well, such as genetics to predict Transcription Factor Binding Sites (TFBS). Garbelini and colleagues developed a Memetic Framework for Motif Discovery (MFMD) that finds sequence motifs using a memetic algorithm [19].

While there has been work focused on extracting motifs, there is little literature on geometrically transformed motifs. Some of the transformed motifs presented in this paper, mainly transpositions and retrogrades, have been explored and been studied in the form of musical symmetry [27][28]. There has also been some work on defining various mathematically transformed versions of motifs and full definitions can be found in [19]. Mathematically transformed motifs have also been defined in the form of several blog posts. Some of the motifs described include The Klein 4-Group and Rachmaninov's transformation of Paganini's Violin Caprice 24 motif in Rhapsody on a Theme of Paganini [20].

3. Background

3.1. Musical Notes

Understanding the workflow of AATMoF requires knowledge of some general music terminology. Most music data are converted to numbers, but to understand the meaning of said numbers some definitions must be in place. There are 7 notes in modern music (A, B, C, D, E, F, G) that can have variations in the form of sharps or flats. These notes are separated by steps which define the distance between different pitches. B and C, and E and F are separated by half steps, and the rest

are separated by whole steps (two half steps). A sharp adds a half step to a note, and a flat subtracts a half step from a note. For example, F# (F sharp) is one half step higher than F, and Bb (B flat) is one half step lower than B. A natural means that just the base note is being played without any sharps or flats [27].

The musical notes are represented in octaves of 8 notes. To define what octave a note is part of, the octave number is written after the note: B2 is the B note in the 2^{nd} octave. A piano generally has 88 notes and 7 Octaves, but for our algorithm we used the 128 possible notes provided by MIDI data [21]. An example of a chart that converts MIDI note numbers to actual notes can be found in Figure 1.

A musical key defines the pitches that form a piece. This is in the form of a scale, or 8 notes that the piece is based off. A scale is a sequence of notes ordered by pitch. Scales usually have lengths of 8 notes or one full octave. A key signature defines the sharps or flats present in a piece, but accidentals -sharps, flats or naturals that are not in the key signature- can occur throughout a musical piece. A minor key is a normal major scale with the third, sixth, and seventh keys being lowered by one half step. The Bach Piece Contrapunctus 7, which we use for our final comparison is in the key of D minor [22].

A time signature describes how many beats are in a measure. A measure is a single unit of time that features a specific number of beats. A motif is the smallest thematic unit that is part of a piece [1]. In Contrapunctus 7, the motif appears in the first two measures and is 8 beats long (14 notes).

```
control_change channel=12 control=101 value=0 time=10
control_change channel=12 control=100 value=0 time=10
control_change channel=12 control=6 value=12 time=10
pitchwheel channel=12 pitch=0 time=10
control_change channel=12 control=93 value=0 time=10
control_change channel=12 control=91 value=64 time=10
control_change channel=12 control=64 value=0 time=10
control_change channel=12 control=11 value=127 time=10
control_change channel=12 control=10 value=64 time=10
control_change channel=12 control=7 value=127 time=10
control_change channel=12 control=1 value=10 time=10
note_on channel=12 note=62 velocity=64 time=1810
note_on channel=12 note=62 velocity=0 time=470
note_on channel=12 note=69 velocity=64 time=10
```

Figure 2: Example of MIDI Messages in Track 6 of Bach's Contrapunctus 7

3.2 MIDI Data

Musical Instrument Digital Interface (MIDI) data is a way to connect devices to create and play sound. This allows different electronic instruments, such as keyboards to communicate with each other. MIDI data can thus be used to save and play pieces. By taking MIDI data as input, a computer, or a program that simulates musical notes, can play a song.

MIDI data mostly comes in the form of MIDI messages which are part of MIDI tracks. Many pieces have several instruments playing at the same time, and generally, the data for the notes played by each instrument is placed on a different track. This varies from piece to piece but was the case for most of the data that we used. Some instruments, such as the piano, use two hands to play simultaneous melodies. Sometimes, as was the case for Contrapunctus 7, each of these hands is recorded on a separate track. This can lead to a piece having 4 tracks when two pianos are playing.

Each MIDI track has MIDI messages that allow keyboards to play specific notes at specific time intervals. Figure 2. shows an example of the MIDI messages at the beginning of a piece. MIDI messages that play notes are identified by either note_on or note_off parameters. A note_on represents a note being pressed on a piano, and a note_off means that a note is being unpressed.

The channel number represents which instrument is being played. The note number represents which note is being played on a scale of 0 to 127. The velocity can be used to find the dynamic of a note. Note that a velocity of 0 is equivalent to a note_off command. The time represents in MIDI time units, the time between this current message and the previous message.

3.3 Mathematical Transformations

There are several mathematical transformations in the two-dimensional plane with translations, reflections, rotations, and dilations being the most used. These transformations are mathematically defined here and are later adapted to a musical context. Figure 4 provides visual illustrations of the four transformations considered in the context of this paper.

3.3.1 Translations

The first major kind of mathematical transformation is a translation. We can view an example of this by defining the function *F* as:

$$F(x, y) = (x + a, y + b) \tag{1}$$

This function translates a point (*x*, *y*) in the cartesian plane to another point (*x'*, *y'*) that is *a* units higher and *b* units to the right of the original point. This is illustrated by Figure 4a with a=2 and b=3.

3.3.2 Reflections

The second kind of mathematical transformation is a reflection. There are two variations on this transformation, y-axis reflections and x-axis reflections. Figure 4b. displays their graphs. The y-axis reflection is described through the following function:

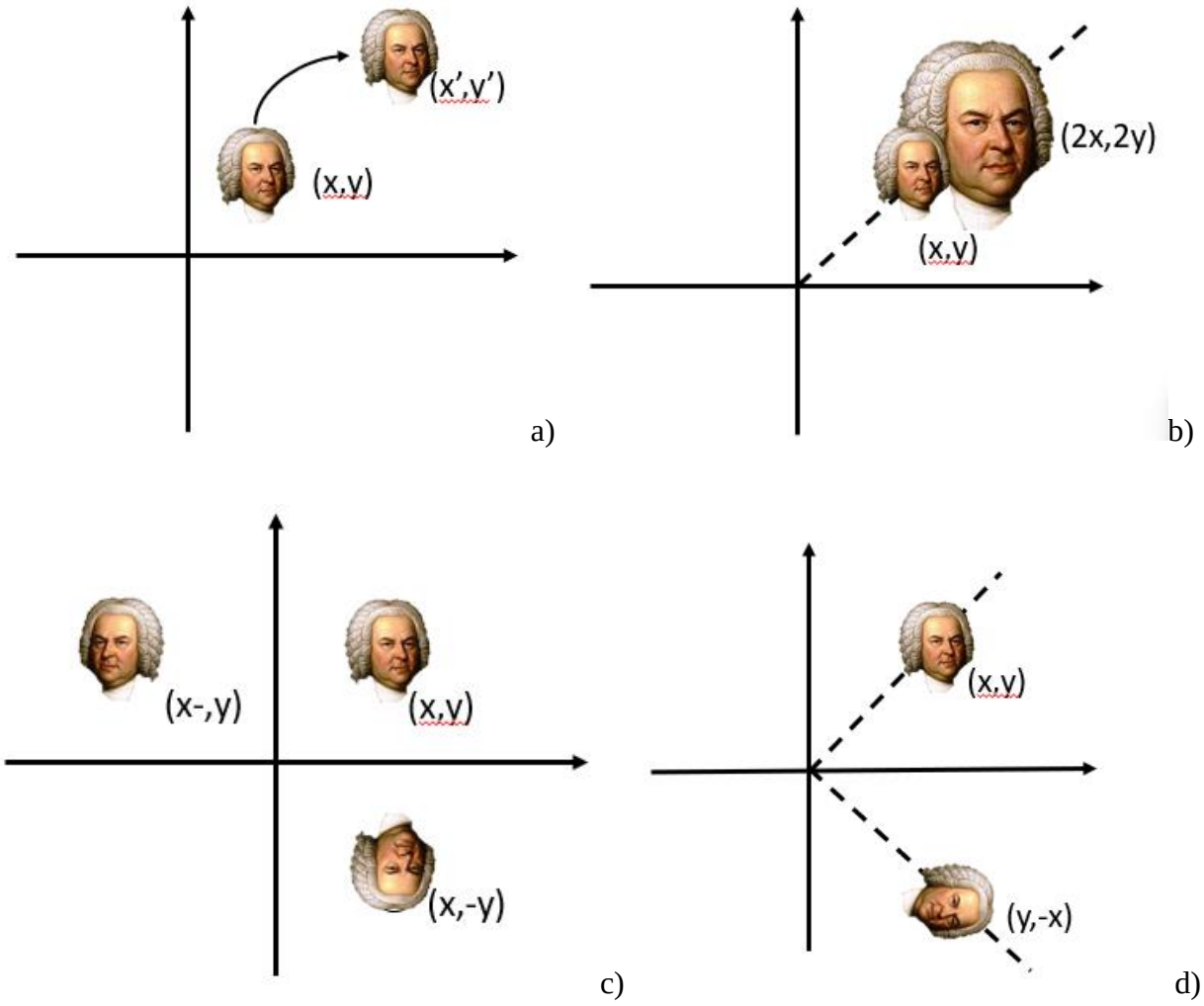

Figure 4. Illustrations for the mathematical transformations a) translation, b) reflection, c) dilation and d) rotation.

$$F(x, y) = (-x, y) \tag{2}$$

while the x-axis reflection is defined as follows:

$$F(x, y) = (x, -y) \tag{3}$$

3.3.3 Dilations

Dilations are described through the following function:

$$F(x, y) = (cx, cy) \tag{4}$$

where *c* is a scalar. Figure 4c shows an example of a dilation where *c*=0.5.

3.3.4 Rotations

Clockwise rotations (illustrated in Figure 4d) are defined through the function:

$$F(x, y) = (-y, x) \tag{5}$$

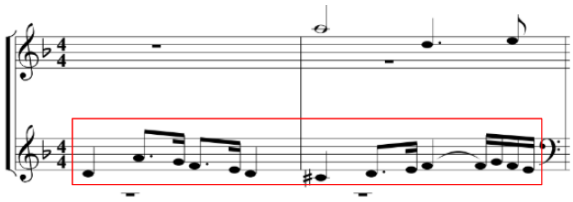

Figure 5. Original Bach Contrapunctus 7 Motif

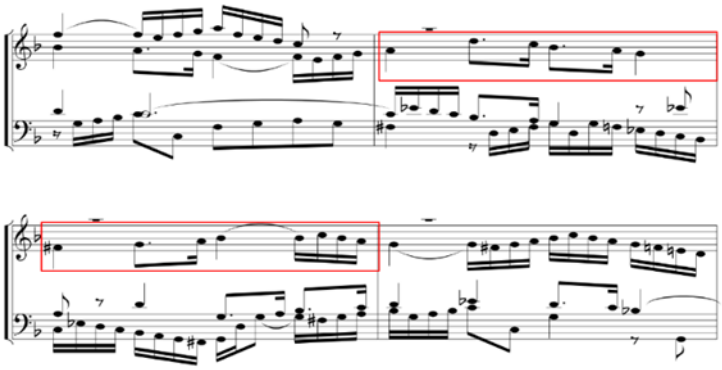

Figure 6. Bach Contrapunctus 7 Motif Pitch Translated Upwards (transposition)

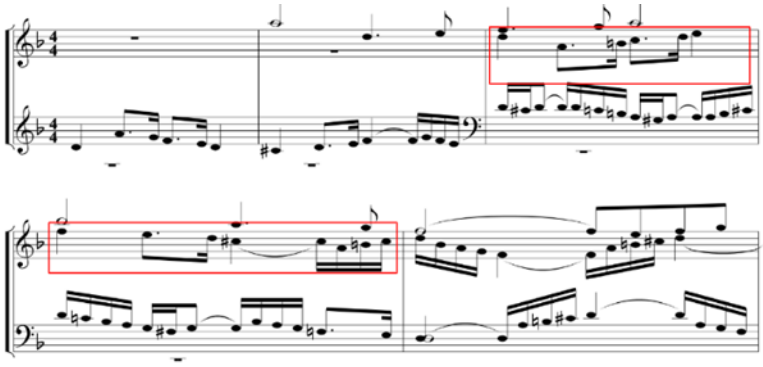

Figure 7. Bach Contrapunctus 7 Horizontally Flipped and Pitch Translated Upwards

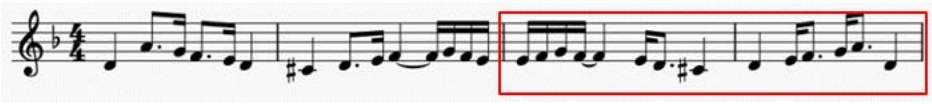

Figure 8. Simulated Bach Contrapunctus 7 Vertical Reflection

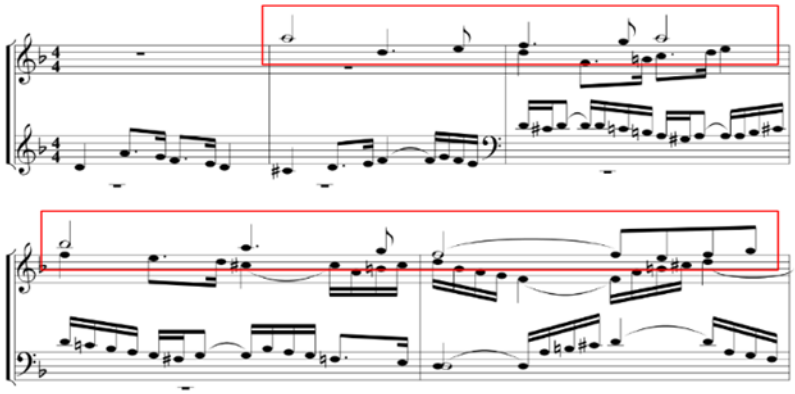

Figure 9. Bach Contrapunctus 7 Motif Time Dilation

3.4 Mathematical Transformations in Music

In this section we describe the applications of the mathematical transformations described previously to a musical context. In Figures 6-9 we give examples of the mathematical transformations of the original Bach Contrapunctus 7 motif, which is showed on Figure 5.

3.4.1 Repetitions and Pitch Shifts

Since there are two kinds of translations, vertical and horizontal, there are also two kinds of motif translations. The first is a horizontal translation where the x axis is viewed as time. This occurs when the motif exactly repeats sometime later in the piece at the same pitch and is known as a repetition in musical context. Neither of the pieces that we analyzed had these types of translations. The musical representation of a vertical translation is a pitch shift, where every note in the motif is increased by a certain number of whole or half steps. This is referred to as a transposition in music theory. The combination of both kinds of translations makes for the most common

mathematical translation of a motif. An example of this combination from Bach's Contrapunctus 7 can be found in Figure 6.

3.4.2 Horizontal Reflections

Horizontal reflections can also be musically represented and occur when every note is flipped over the central staff line. These are referred to as retrogrades and occur regularly in Bach's Contrapunctus 7; an example can be found in Figure 7.

3.4.3 Vertical Reflections

Vertical reflections happen when a musical motif is flipped across a horizontal axis, like a staff line. These are called inversions in music theory. These did not occur in any of the pieces we observed but a simulated vertical reflection of the motif from Bach's Contrapunctus 7 can be found in Figure 8.

3.4.4 Time Dilations

The musical representation of a cartesian dilation appears in the form of a time dilation where every note's length is scaled in response to some constant. Usually this is just in the form of a motif being played at half or double speed and was found in Bach's Contrapunctus 7 as seen in in Figure 9.

4. Methods

4.1 Software Components

Figure 10. AATMoF Workflow

AATMoF was developed completely in the Python programming language. Python is an object-oriented high-level programming language that is widely used by consumers [23]. We choose Python due to the already created MIDI conversion tools and to have the ability to access its libraries. For this project we used Python version 3.8.10. In order to make it easy to add notifications we developed AATMoF in a Jupyter notebook and published it on GitHub [24].

Jupyter notebook is a web-based interactive computing platform and effectively allows users to run only small sections of a program instead of the whole file [25]. We used Jupyter notebook version 6.4.3 [26].

4.2 Program Workflow

AATMoF takes in as input a user-submitted MIDI data file, a start and end in array format (described in 4.2.1) to define the base motif, a track to specify where AATMoF should get the base motif from and a track to specify where the algorithm should search for transformed instances of the motif. The algorithm takes these data, processes them, transforms the original motif, and searches for repeated instances of these transformed motifs in the musical piece. The entire workflow can be seen in Figure 10.

4.2.1 Data Processes

AATMoF's first task is to convert the raw MIDI data files into a format that can be searched for motifs. AATMoF then takes the raw MIDI messages and takes out every message that is not a note_on or note_off message since those are beyond the scope of our research. Using these note_on and note_off messages, AATMoF builds of a list of triples to represent each note in the form (1/0 if On or Off, Note Number, Time since previous message) as seen in Figure 11a. AATMoF also creates a list of binary times for each note press in case the user needs a timeline to recognize time dilations (Figure 11b).

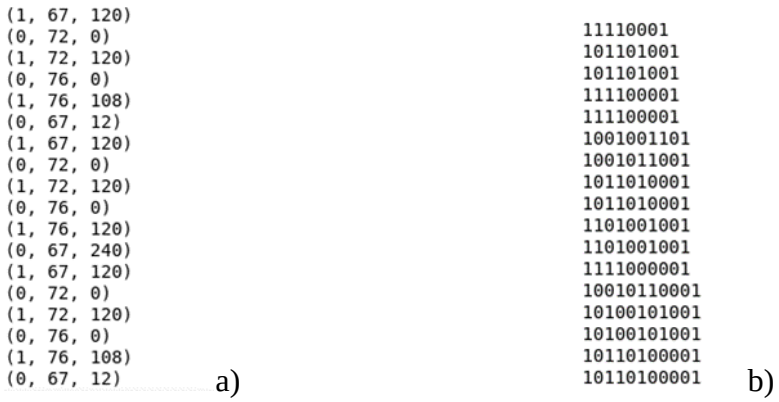

Figure 11. a) Part of a musical piece in triple format b) Chronological time values for each event (note press/removal).

Following this, we used the song format representation (here referred as array format) described in [18] to analyze the pieces. This method takes snapshots of the instrument each time something is updated. For example, if a key is pressed, there will be a new array in the representation. When a finger is lifted off the key, there will be another array reflecting the fact that there is no key currently pressed. If a note is being played at that specific instance (or array), the note's place in the array will have a value of 1. If the note is not being played, it will have a value of 0. An example of two notes being played in the array format can be seen in Figure 12. As shown, the two notes being played (in the red circles) are represented by 1s. When each of the

notes stops being played, a new empty array is created to represent that. The entire track can be represented using these arrays and they are all that is needed for the transformed motif search.

4.2.2 Motif Comparison

As mentioned previously, the user needs to insert a motif start and motif end value so that the program knows what the original motif is. These values are in arrays and since there are generally two arrays for each note (one for the note being pressed, a second for when the note finishes playing), the difference between these is usually twice the length of the motif in notes. Incidentally, slurs and ties do result in a slight problem since two notes that are tied can be written in 2 arrays.

After the motif is defined, the program first finds the maximum pitch translation up, and pitch translation down by taking the maximum and minimum MIDI notes and subtracting them from the highest and lowest pitched notes in the motif. Using these two values, AATMoF finds all possible upward and downward shifts in motifs and concatenates all the arrays. AATMoF then compares this large array with arrays of the same size from the original piece by taking a group of consecutive notes from the piece, concatenating their arrays, and subtracting the two large arrays. The sum of the remaining values gives the number of differences between the musical segments where 0 is the best (no difference) and 2 times the motif length is the worst (every note is different between the two segments. By shifting through every group of consecutive notes in the piece with the same length as the motif, the algorithm checks the entire piece for all pitch shifted versions of the motif and gives a difference value for every comparison. These can then be filtered for only values that have a good chance of being transformed versions of the motif (further discussed in results).

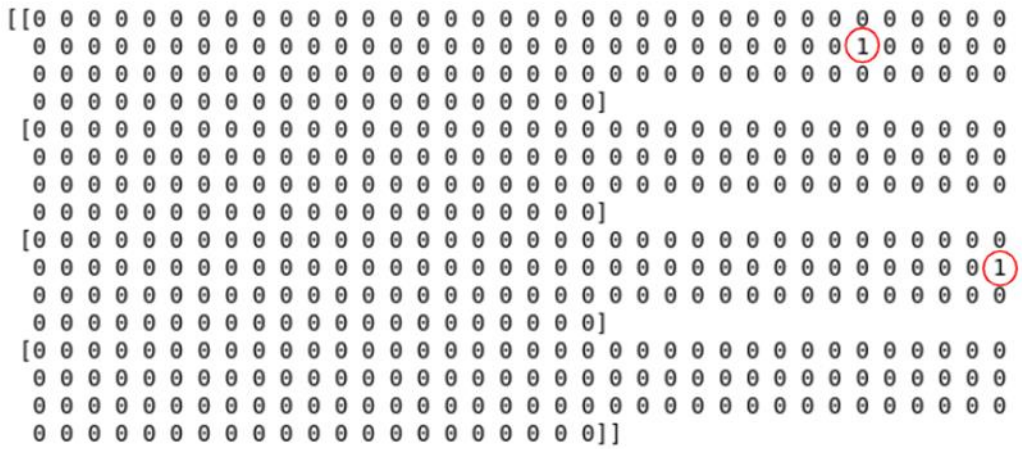

Figure 12. Array Representation of two notes (in red circles) being played.

The user can also input a horizontally flipped version of the base motif and the algorithm can also search the piece for every version of this transformed motif using the same method as above. Since the array format described in 4.2.1 does not look at the time values, time dilations of either the horizontally flipped motif or the original motif are also found using this method. If the user desires, they can analyze the binary time list and compare the times between notes in the original motif and in each of the suspected time dilations. An example of the output for pitch translated versions of the original motif in Bach Contrapunctus 7 is shown in Figure 16.

The algorithm outputs the distance (up/down) in half steps from the original motif and displays the note numbers for the beginnings of each of the transformed versions of the motif. The algorithm also outputs the difference between the phrase that starts on the note number and the pitch translated version of the motif that the half step count corresponds to.

5. Results

We compared an AATMoF analysis to human analysis of two pieces. We tested AATMoF on Contrapunctus 7, and Prelude in C, by Bach. Both pieces were first analyzed by humans; Contrapunctus 7 had 3 human analyses and the best analysis (defined on correctness and motif

frequency) was used for comparison. The human who completed the best analysis then also analyzed Prelude in C. Figures 17 and 18 display a segment of each of the two final human analyses. Both pieces were then machine analyzed and had one human analyze the data produced and check for accuracy. Motifs with a difference rating of below 10 were considered as transformed motifs from Contrapunctus 7 and motifs with a difference rating of 0 were considered as transformed motifs for Prelude in C due to the similarity in musical structure.

```
4 246 4
4 252 8
4 306 10
4 312 8
4 318 10
---
---
6 362 8
6 368 8
6 374 8
6 380 8
6 386 8
6 392 10
6 596 10
6 602 8
6 608 10
6 692 10
6 698 8
```

Figure 13. Example of AATMoF Output in the form: Number of half steps above/below the motif, the note number of the start of the suspected transformed version of the motif, the difference with the transformation of the original motif.

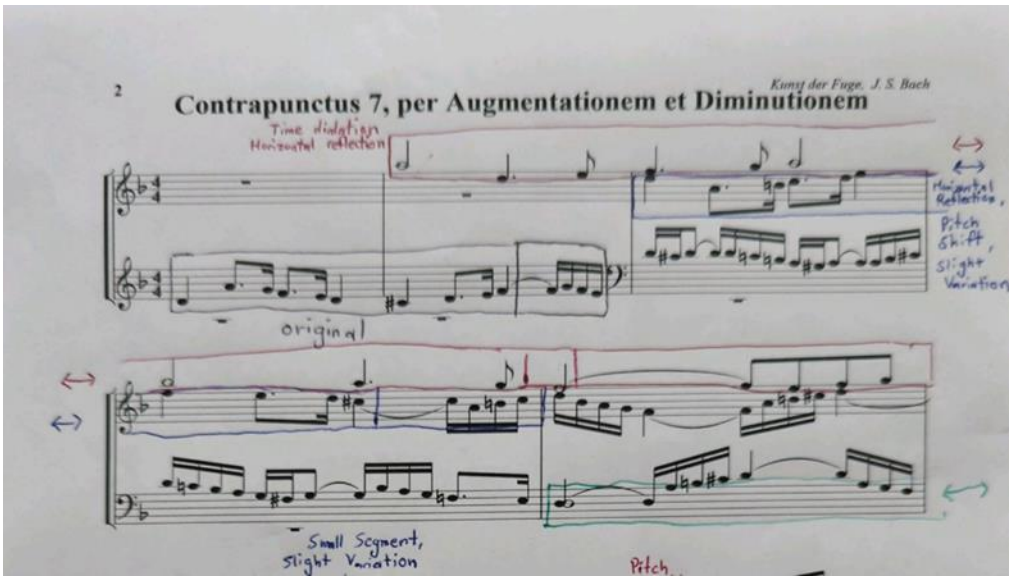

Figure 14. Contrapunctus 7 human analysis.

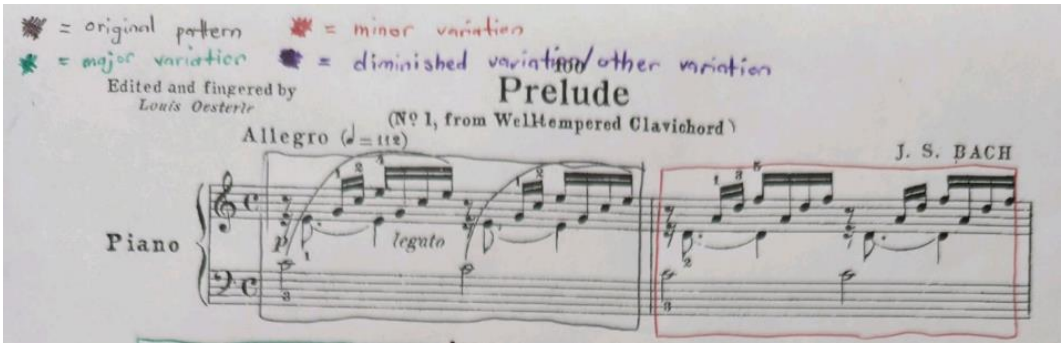

Figure 15. Prelude in C human analysis

For Contrapunctus 7 the Human Analysis found 28 versions of the motif. The machine analysis found 33 transformed versions of the motif when the difference level (or the metric described in the introduction) was set to be less than 10. This means that only phrases with less than 10/2=5 differences were considered transformed versions. The machine identified 27 out of the 28 human-detected motifs. Some of the new motifs identified by the machine may be considered to not be full motifs but it is still interesting to see that AATMoF was able to detect most of the human-detected motifs. An example of one of these partial motifs is shown in Figure 16. For Prelude in C, the human found 3 versions of the motif and no transformations and

AATMoF found 5 transformed versions of the motif. In this case AATMoF was correct and these 2 extra motifs were in fact motifs. Since most of the piece looks very similar it can get difficult for humans to differentiate between slight variations and actual transformed motifs especially when there are pitch changes. Figure 17. Shows the original motif in Prelude in C and Figure 18. Shows the transformed version that was marked as a slight variation. AATMoF found these motifs much faster than the human analysis. It took the human 7 hours for Contrapunctus 7 and 30 minutes for Prelude in C. The total time in running all parts of AATMoF was less than 10 seconds for both pieces but varied depending on background processes in the computer.

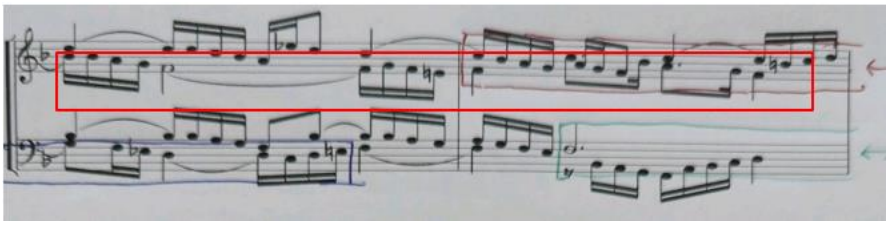

Figure 16. Example of Partial Motif Found by AATMoF

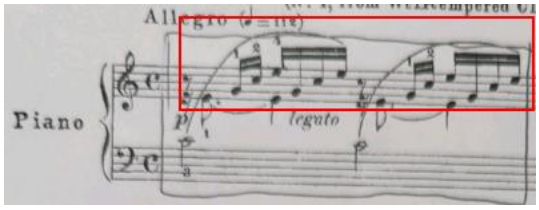

Figure 17. Original Motif in Prelude in C

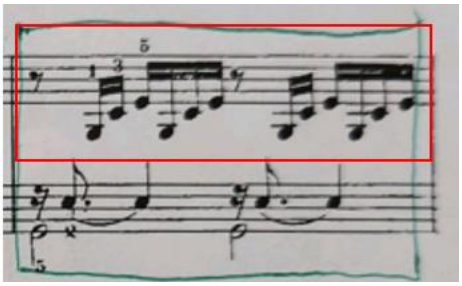

Figure 18. Missed Prelude in C motif

6. Conclusions and Future Work

All aspects of AATMoF were successful in running and AATMoF did produce results that were similar to the human analysis. In addition, AATMoF found these transformed instances much faster than the human analyses. There are still several things that can be improved and developed in future work. First, having AATMoF automatically generate the horizontally transformed motif, and automatically classify time dilation motifs would be very beneficial.

Finding motifs in music is important for musical analysis and could have applications in copyright law. In addition, a musician could use the motif structure generated by AATMoF analysis to compose original pieces in the same style as the author whose work is being analyzed. We plan to investigate these ideas in future research.

7. Bibliography

[1] J. D. White and J. White, "The analysis of music." Metuchen, NJ: Scarecrow Press, 1984.

[2] R. J. Cason and D. Müllensiefen, “Singing from the same sheet: computational melodic similarity measurement and copyright law,” *International Review of Law, Computers & Technology*, vol. 26, no. 1, pp. 25–36, 2012.

[3] I. Stav, “Musical plagiarism: a true challenge for the copyright law,” *DePaul J. Art Tech. & Intell. Prop. L*, vol. 25, p. 1, 2014.

[4] T. Anders, “A model of musical motifs,” in *International Conference on Mathematics and Computation in Music*, 2007, pp. 52–58.

[5] A. Cardoso, T. Veale, and G. A. Wiggins, “Converging on the divergent: The history (and future) of the international joint workshops in computational creativity,” *AI magazine*, vol. 30, no. 3, pp. 15–15, 2009.

[6] J. Armstrong, “The Formation of Bach’s Motifs: Chapter Two of André Pirro’s" L’Esthétique de Jean-Sébastien Bach"," *BACH: Journal of the Riemenschneider Bach Institute*, vol. 41, no. 1, pp. 32–96, 2010.

[7] A. L. Robin, “Motive and motif in the church music of Johann Sebastian Bach,” *Theology today*, vol. 63, no. 1, pp. 38–47, 2006.

[10] R. Benammar, C. Largeron, V. Eglin, and M. Pardoen, “Discovering motifs with variants in music databases,” in *International Symposium on Intelligent Data Analysis*, 2017, pp. 14–26.

[11] A. Jiménez, M. Molina-Solana, F. Berzal, and W. Fajardo, “Mining transposed motifs in music,” *Journal of Intelligent Information Systems*, vol. 36, no. 1, pp. 99–115, 2011.

[12] O. Lartillot, “Perception-based advanced description of abstract musical content,” in *Digital Media Processing for Multimedia Interactive Services*, World Scientific, 2003, pp. 320–323.

[13] J. Yaconelli and R. M. Keller, “Discovery and Utilization of Jazz Motifs for Computer-Generated Solos,” *Computer Simulation of Musical Creativity*, 2018.

[14] A. Laaksonen and K. Lemström, “Transposition and time-warp invariant algorithm for detecting repeated patterns in polyphonic music,” in *6th International Conference on Digital Libraries for Musicology*, 2019, pp. 38–42.

[15] B. Meudic, “Musical pattern extraction: From repetition to musical structure,” *Computer Music Modelling and Retrieval, Montpellier, France*, 2003.

[16] B. Janssen, “Discovering repeated patterns in music: potentials, challenges, open questions,” *Sound, Music, and Motion: 10th International Symposium*, 2013, p.277-297.

[17] D. Pituk, "Automatic audio sample finder for music creation: Melodic audio segmentation using DSP and machine learning," 2019.

[18] F. T. Liang, M. Gotham, M. Johnson, and J. Shotton, “Automatic Stylistic Composition of Bach Chorales with Deep LSTM.,” in *ISMIR*, 2017, pp. 449–456.

[19] J. M. Caldonazzo Garbelini, A. Y. Kashiwabara, and D. S. Sanches, “Sequence motif finder using memetic algorithm,” *BMC bioinformatics*, vol. 19, no. 1, pp. 1–13, 2018.

[20] “Motifs: Molecules of Music,” *ThatsMaths*, May 31, 2018. https://thatsmaths.com/2018/05/31/motifs-the-molecules-of-music/ (accessed Sep. 25, 2022).

[21] J. Hass, “MIDI Chapter Three: MIDI Data Format,” *Introduction to Computer Music*. https://cmtext.indiana.edu/MIDI/chapter3_midi_data_format.php (accessed Sep. 25, 2022).

[22] “Minor Key Signature,” *Essential Music Theory*. https://www.essential-music-theory.com/minor-key-signature.html (accessed Sep. 25, 2022).

[23] “Python.org,” *Python.org*. https://www.python.org/ (accessed Sep. 25, 2022).

[24] “GitHub: Where the world builds software,” *GitHub*. https://github.com/ (accessed Sep. 25, 2022).

[25] T. Kluyver *et al.*, "Jupyter Notebooks-a publishing format for reproducible computational workflows.," ICEP, vol. 2016. 2016.

[26] “Project Jupyter.” https://jupyter.org (accessed Sep. 25, 2022).

[27] Colleen Duffy, “Symmetrical musical pieces”, http://sheenabumangil.blogspot.com/2011/03/what-is-motion-geometry.html (accessed Sep. 25, 2022).

[28] John H. Conway, Heidi Burgiel, and Chaim Goodman-Strauss, “The Symmetries of Things,” A K Peters 2008.

[29] V. Hart, “Symmetry and Transformations in the Musical Plane,” *Proceedings of Bridges 2009: Mathematics, Music, Art, Architecture, Culture*, Pages 169–176.