Research Article

Temporal & Dynamics Analysis System for Piano Performances

Daniel Jackson, Department of Electronic Engineering, University of York, United Kingdom

Abstract

An automated music performance analysis system targeted at analysing and comparing the tempo and dynamics of two piano performances is outlined in this article. The main application of this system is for helping intermediate pianists learn to play more expressively. The tempo detection algorithm utilised in the system calculates the spectral flux of a signal and then uses enhanced autocorrelation and a set of rules to pick the best BPM. The dynamics analysis uses MIDI velocity for analysis of MIDI dynamics, and the RMS amplitude for analysis of audio signals. The tempo detection achieved good results in tests, achieving a Mean Absolute Error of 7.92 BPM for audio files and 7.12 BPM for MIDI files. The dynamics analysis performed reasonably for MIDI signals, but poorly for audio signals and would need significant further development in future iterations. A user interface was also developed for ease of use. Background literature is studied and reviewed before the full system design and implementation is detailed, and a detailed system analysis and discussion are then undertaken.

Word Count: 4302

1. INTRODUCTION

There is evidence of discussion and observation surrounding music performance dating back to the 1700s (Gabrielsson, 2003). Despite this, it did not begin being formally and empirically measured until around the turn of the 20th century. Based on a study by Gabriellson (1999), the most common type of Music Performance Analysis (MPA) research done is the measurement of performance parameters, with the most popular parameters being timing and dynamics. This is likely due to tempo being considered the key interpretive element in a performance (Bowen, 1996), as well as having a profound impact on the perception of emotion in a musical performance (Gabriellson, 1996).

This article will detail the design, implementation and analysis of a software system which performs automated temporal and dynamics analysis of an audio and MIDI recording. The audio file would be a reference recording, such as a professional performance, and the MIDI file would be the user's performance. The professional audio recording and the user's MIDI performance are then compared. Background information from relevant literature will also be considered to see how this system fits in with current and previous research.

1.1 Background

Tempo can be defined and measured in many ways. It is common to measure both the global (mean) tempo of a piece, and the local tempo over smaller sections. Dynamics are also commonly considered one of the most expressive performance variables (Gabriellson, 2003), and are interesting to consider as they are relatively poorly defined within musical scores. No absolute values are given, with values usually ranging from ***pp*** (very quiet) to ***ff*** (very loud) (Lerch, 2012), leaving it up to the performer to interpret exactly what this means.

Early temporal analysis of music performance commonly relied upon manual 'foot-tapping' along with the perceived tempo of the piece. Alternatively, Acoustic Onset Times (AOTs) of notes could be manually measured which could then be used to calculate local tempos from the Inter-Onset Intervals (IOIs), the distance between each note's onset. Automatic tempo detection and beat tracking first began being mentioned in literature in the 1980s with the advent of Digital Signal Processing (DSP), such as in the paper by Chafe, Mont-Reynaud & Rush (1982). This continued throughout the 1990s in papers such as Allen & Dannen (1990).

There are many modern approaches to automated tempo detection using DSP techniques. One common approach outlined in Lerch (2012) is to automatically detect the AOTs of each note, then use this to determine the positions of beats and finally calculate the Inter-Beat Intervals (IBIs). These IBIs can then be used to calculate the tempo according to Equation 1 (Lerch, 2012), where $T_{local}(k)$ is the local tempo at beat k and $t_b(k)$ is the of beat k.

$$T_{local}(k) = \frac{60}{t_b(k+1) - t_b(k)} \tag{1}$$

The global tempo can then be calculated as the mean of this value over a section, as shown in Equation 2, where N is the total number of beats in the section.

$$T_{global} = \overline{T_{local}(k)}\ for\ 0 < k \le N \tag{2}$$

The key to this approach is accurately detecting the AOTs of notes. This is done using a novelty function which is used to determine the amount of new information in each analysis frame, and then using a peak-picking algorithm to determine the largest maxima of the output of the novelty function (Lerch, 2012). There are many different novelty function implementations. Some common implementations include calculating the spectral flux of the signal (Laroche, 2003) or calculating a weighted High Frequency Content (HFC) transformation of the signal (Bello et al, 2005).

Spectral flux measures the change in spectral shape of a signal and is defined as the average difference between STFT frames (Lerch, 2012). Equation 3 shows a spectral flux implementation for detecting acoustic onsets, where $X(f, t_i)$ is the Fast Fourier Transform (FFT) at sample point i and $G(x)$ is a nonlinear compression function, commonly implemented as $G(x) = \sqrt{x}$ (Laroche, 2003).

$$E(i) = \sum_{f=f_{min}}^{f_{max}} G(|X(f, t_i)|) - G(|X(f, t_{i-1})|) \quad (3)$$

Another method of automatically detecting the tempo also makes use of the spectral flux signal. First, the spectral flux of the audio signal is calculated, and then two time-stretched duplicates of this signal are created, stretched by factors of 2 and 4 (Percival & Tzanetakis, 2014). The Autocorrelation Function (ACF) is then applied to all three versions of the spectral flux. The ACF detects periodicity in signals, and therefore can be used to detect the Beats per Minute (BPM) of music with a regular pulse. Calculating the ACF of all three spectral flux signals and then summing them enhances the peaks from the ACF (shown in Figure 1). This makes it is easier to pick out the most likely BPMs, with the largest peaks usually falling at the actual BPM or integer multiples of the BPM (Percival & Tzanetakis, 2014). The most likely of these BPMs can then be chosen by evaluating them against a series of impulse trains, applying a Gaussian window to the data, and evaluating the estimated BPM to determine whether the actual value is an integer multiple of the detected BPM. In Percival & Tzanetakis (2014), this evaluation was done using a decision tree trained through machine learning.

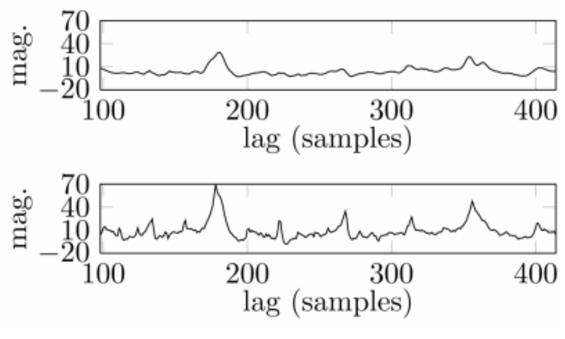


Figure 1: Top: Autocorrelation of the spectral flux of the audio signal. Bottom: autocorrelation of the summed spectral fluxes of the original audio signal and the two time-stretched versions (Percival & Tzanetakis, 2014).

Measuring the intensity of an audio signal is a simple, albeit sometimes flawed, method for analyzing the dynamics in a piece of music. The Root Mean Squared (RMS) amplitude of an audio signal is one of the most commonly analyzed intensity features (Lerch, 2012), and is calculated over a section of an audio signal using Equation 4, where N is the number of samples and $x(n)$ is the amplitude of the signal at sample point n.

$$A_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x(n)^2} \quad (4)$$

Dynamics can be more easily and accurately analyzed from MIDI files. This is because velocity is a part of the MIDI standard definition and defines how hard a note was triggered (MIDI Manufacturers Association, 1996). Mappings between MIDI velocities and dynamics have been made in literature such as Berndt & Hähnel (2010) and in The Complete MIDI 1.0 Detailed Specification (MIDI Manufacturers Association, 1996). One example is shown in Table 1.

Dynamic	*ppp*	*pp*	*p*	*mp*	*mf*	*f*	*ff*	*fff*
Velocity	2	36	48	64	83	97	111	125

Table 1: Table showing mappings between MIDI velocity and musical dynamics (Berndt & Hähnel, 2010).

2. SYSTEM DESIGN & IMPLEMENTATION

The developed system is targeted at intermediate pianists who wish to play pieces more expressively and closely match the performances of professional recordings. To achieve this, the system accepts one MIDI file and one audio recording, then performs analysis on each of these separately. Detailed comparative feedback is then provided to the user. The system is written in MATLAB and makes use of the Signal Processing Toolbox and the MIDI Toolbox (Eerola & Toiviainen, 2016).

First, the audio is pre-processed to get it into an appropriate state for analysis. If it is a stereo recording, it is converted to mono by summing the audio channels, and then normalized between -1 and 1. Finally, any leading and trailing silence is cropped from the file.

The global tempo of the audio file is then estimated, and this is used to divide the audio into segments for which the local tempo can be calculated. These segments are all approximately 2 bars in length based on the detected global tempo and given time signature. The MIDI file is then synthesized into audio using a function from the MIDI toolbox, and the same analysis is performed on this to extract the global and local tempos.

The tempo detection is mostly done using the autocorrelation method described in section 1 of this article, with some modifications. The spectral flux is calculated using a STFT with a 512-sample window length, 0 samples of overlap and 2048 frequency bins. Only the frequency bins which fall within the range of 100 Hz and 10 kHz are used in the calculation. Once the spectral flux across the audio has been calculated, two copies of the spectral flux are taken and are time-stretched by factors of 2 and 4. This is achieved using a 256-point Hann window with the MATLAB function *stretchAudio*. The autocorrelation of each of these spectral flux signals is then calculated and summed, and the largest 10 peaks are picked from it. The mean and median of these values are then picked, and a check is run to determine if the mean value falls within ±10% of twice or half the median value. If it does, the mean is either doubled or halved to give an approximate BPM. This BPM is then used to calculate segment lengths, split the audio into segments, and then analyze the local BPMs. The final global BPM is then calculated as the mean of all the local BPMs.

The dynamics of the audio clip are then analyzed by calculating the RMS value of the entire audio signal in 500ms frames. An example plot of this is shown in Figure 2. The RMS value across each segment is then averaged to determine the average RMS value for each segment.

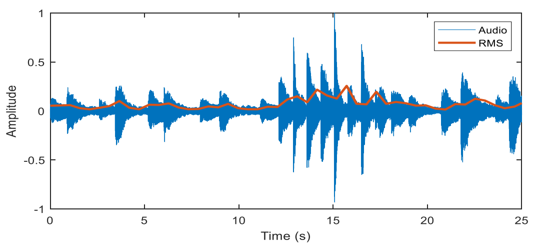


Figure 2: Section of an audio waveform with its corresponding RMS value plotted over the top.

This average value is then compared against a range of RMS values, shown in Table 2, to determine what dynamic they should be categorized as, from a minimum value of ***pp*** to a maximum value of ***ff***. These values were based on the MIDI to dynamics mapping found in Berndt & Hähnel (2010) and shown in Table 1. However, the ***ppp*** and ***fff*** values were removed, and RMS values were then calculated by recording a MIDI piano using these velocities and then measuring the RMS of the recording.

Dynamic	Velocity	RMS
pp	1	0.0549
p	48	0.0718
mp	64	0.1026
mf	83	0.1524
f	97	0.2184
ff	111	0.2997

Table 2: MIDI velocity and RMS dynamic mappings used in the system.

A similar method is used for the analysis of the MIDI velocity: each segment is looped over; the average velocity is found from each segment; and then this average is compared against the velocity values shown in Table 2. Once both the audio and MIDI files have had their dynamics analyzed, they can be plotted on the user interface (UI) over time as shown in Figure 3.

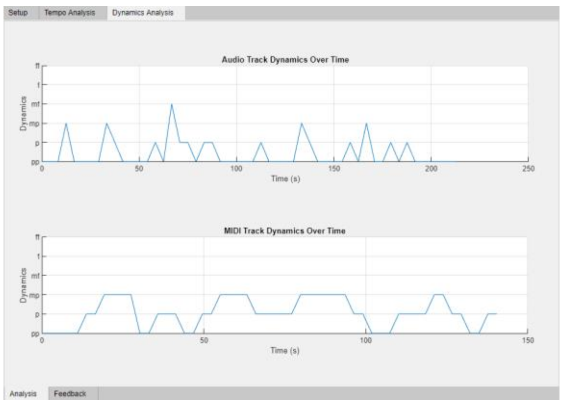
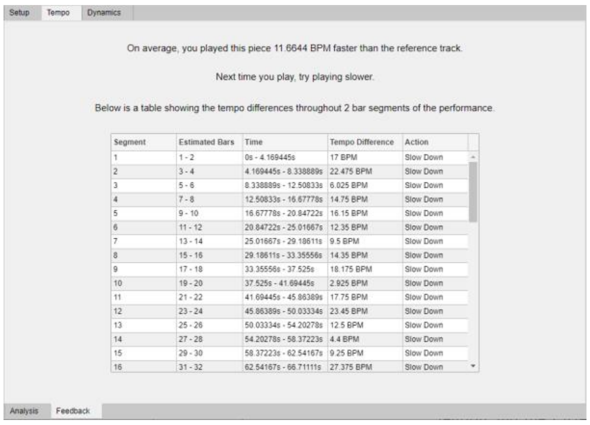


Figure 3: Screenshot of the dynamics analysis UI with the dynamics over time plotted of the MIDI and audio track.

Once the analysis is completed, the values from the audio and MIDI files can be compared. For tempo this is done by subtracting the local tempos of the audio file from the local tempos of the MIDI file. These differences are then displayed in a table, along with a statement to the user as to whether they need to play faster or slower in each segment. There is also a high-level statement made to the user for the overall performance. An example of this UI page is shown in Figure 4.



On average, you played this piece 11.6644 BPM faster than the reference track.

Next time you play, try playing slower.

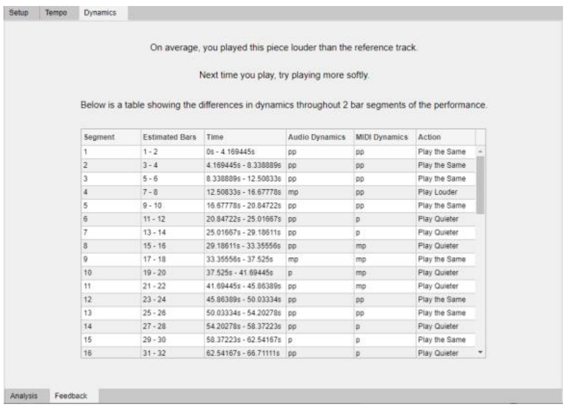
Below is a table showing the tempo differences throughout 2 bar segments of the performance.

Segment	Estimated Bars	Time	Tempo Difference	Action
1	1 - 2	0s - 4.169445s	17 BPM	Slow Down
2	3 - 4	4.169445s - 8.338889s	22.475 BPM	Slow Down
3	5 - 6	8.338889s - 12.50833s	6.025 BPM	Slow Down
4	7 - 8	12.50833s - 16.67778s	14.75 BPM	Slow Down
5	9 - 10	16.67778s - 20.84722s	16.15 BPM	Slow Down
6	11 - 12	20.84722s - 25.01667s	12.35 BPM	Slow Down
7	13 - 14	25.01667s - 29.18611s	9.5 BPM	Slow Down
8	15 - 16	29.18611s - 33.35556s	14.35 BPM	Slow Down
9	17 - 18	33.35556s - 37.525s	18.175 BPM	Slow Down
10	19 - 20	37.525s - 41.69445s	2.925 BPM	Slow Down
11	21 - 22	41.69445s - 45.86389s	17.75 BPM	Slow Down
12	23 - 24	45.86389s - 50.03334s	23.45 BPM	Slow Down
13	25 - 26	50.03334s - 54.20278s	12.5 BPM	Slow Down
14	27 - 28	54.20278s - 58.37223s	4.4 BPM	Slow Down
15	29 - 30	58.37223s - 62.54167s	9.25 BPM	Slow Down
16	31 - 32	62.54167s - 66.71111s	27.375 BPM	Slow Down

Figure 4: Screenshot of the tempo comparison UI page providing feedback to a user on their performance of a piece.

Comparison between the MIDI and audio dynamics are provided in a similar fashion. The dynamic values are mapped to a value from 1 to 6 (representing ***pp*** to ***ff***) and then these dynamic values for the audio file are subtracted from the dynamic values for the MIDI file. There is then a check to see if the result of this is negative; if it is, it will inform the user to play more quietly, otherwise it will inform them to play more loudly. An example of the UI page providing this feedback is shown in Figure 5.

The UI was designed to give the user detailed, meaningful feedback and analysis of their performances, whilst also being intuitive to use and easy to digest. Whilst it could be argued that too much information is provided for the average intermediate pianist, the high-level statements and simple suggestions make it easy to determine what actions need to be taken to alter the performance.



On average, you played this piece louder than the reference track.

Next time you play, try playing more softly.

Below is a table showing the differences in dynamics throughout 2 bar segments of the performance.

Segment	Estimated Bars	Time	Audio Dynamics	MIDI Dynamics	Action
1	1 - 2	0s - 4.169445s	pp	pp	Play the Same
2	3 - 4	4.169445s - 8.338889s	pp	pp	Play the Same
3	5 - 6	8.338889s - 12.50833s	pp	pp	Play the Same
4	7 - 8	12.50833s - 16.67778s	mp	pp	Play Louder
5	9 - 10	16.67778s - 20.84722s	pp	pp	Play the Same
6	11 - 12	20.84722s - 25.01667s	pp	p	Play Quieter
7	13 - 14	25.01667s - 29.18611s	pp	p	Play Quieter
8	15 - 16	29.18611s - 33.35556s	pp	mp	Play Quieter
9	17 - 18	33.35556s - 37.525s	mp	mp	Play the Same
10	19 - 20	37.525s - 41.69445s	p	mp	Play Quieter
11	21 - 22	41.69445s - 45.86389s	pp	mp	Play Quieter
12	23 - 24	45.86389s - 50.03334s	pp	pp	Play the Same
13	25 - 26	50.03334s - 54.20278s	pp	pp	Play the Same
14	27 - 28	54.20278s - 58.37223s	pp	p	Play Quieter
15	29 - 30	58.37223s - 62.54167s	p	p	Play the Same
16	31 - 32	62.54167s - 66.71111s	pp	p	Play Quieter

Figure 5: Screenshot of the dynamics comparison UI page providing feedback to a user on their performance of a piece.

3. SYSTEM ANALYSIS & DISCUSSION

The system's accuracy and success will now be measured and analyzed, starting with the accuracy of the underlying algorithms.

The success of the tempo detection algorithm will be assessed for both audio files and MIDI files. For audio files, the tempo detection will be compared to the commercially available tempo detection available in Ableton Live 9, for the same piece of audio. This was done for five pieces of audio: Eric Satie's *Gymnopédie No. 1* and *Gnossienne No. 1*, performed by Anne Queffelec; Aphex Twin's *Avril 14th*; and two clips of a chord sequence played on a piano, with each note falling on the crotchet beat of each bar. All these files are included within the supporting files of this article. Inclusion of these files falls under an academic fair-use copyright exemption (Intellectual Property Office, 2014). For MIDI files, the tempo detection analysis will be carried out by using a quantized MIDI performance and then comparing the detected tempo with the actual MIDI tempo. MIDI files for each piece are also included within the supporting files of this article.

Table 3 shows the BPM detection of these audio files compared with Ableton Live's BPM detection of the same file.

Piece	Ableton BPM	MATLAB BPM
Gymnopedie 1	84.34	86.58
Gnossienne 1	95.64	103.86
Avril 14th	78.96	104.23
Strict Crotchets	102	101.38
Lenient Crotchets	102	105.23

Table 3: Table showing the detected BPMs of the audio files from Ableton Live and the MATLAB algorithm.

This shows a reasonably good accuracy for the tempo detection algorithm, with a Mean Absolute Error (MEA) of 7.92 BPM. Three of these results have an error of less than 5%, and another with an error of less than 10%. The highest error is 32% for *Avril 14th*, and the lowest error is 0.61% for the Strict Crotchets recording. If the tempo detection of *Avril 14th* is discounted, the MEA of the system drops to just 3.58 BPM. This implies that the system has good audio tempo detection accuracy in most cases but can be significantly less accurate in a small number of cases. However, significantly more recordings would need to be analyzed to make a definitive statement on the accuracy of this aspect of the system.

Table 4 shows the BPM detection for the MIDI files compared with the actual BPM of the MIDI performance. This shows a similar accuracy to the tempo detection of the audio files, with an MEA of 7.12 BPM, despite the error being better or worse for different pieces. Two of these results have an error of less 5% and a further two have an error of less than 10%. This time *Gymnopedie 1* has the largest error of 18.36%, and *Avril 14th* has the lowest error of 1.19%. If the tempo detection of *Gymnopedie 1* is discarded, the MEA drops to just 4.31 BPM.

Piece	Actual BPM	MATLAB BPM
Gymnopedies 1	100	81.64
Gnossiennes 1	96	89.08
Avril 14th	84	83
Strict Crotchets	102	97.86
Lenient Crotchets	102	107.18

Table 4: Table showing the detected BPMs of the MIDI files and the corresponding actual BPMs of the performance.

A similar accuracy would be expected between the audio and MIDI tempo detection due to them both using the same underlying algorithm. However, discrepancies are to be expected due to the additional synthesis of the MIDI into audio and differences in the MIDI performances compared to the audio performances. And again, for a definitive statement to be made on the success of this element of the system, more MIDI files would need to be analyzed.

The tempo detection algorithm was successful, giving results of a good accuracy for both MIDI and audio, but was significantly challenging to develop. The largest challenge was avoiding integer multiples of the BPM from being chosen, and this is the reason the highest peak from the autocorrelation function is not simply chosen as the BPM. There are also significant improvements that could still be made, particularly for the tempo analysis of the MIDI file. The MIDI analysis relies upon first synthesizing the MIDI into audio and then analyzing this synthesized audio, which is a computationally slow procedure, particularly on longer audio files. Running on a Windows 10 PC with an Intel i7 8700k processor and 16 GB RAM, this process takes approximately 550 milliseconds for a MIDI file approximately 2 minutes long. In future iterations it would be ideal to perform this temporal analysis without having to synthesize the MIDI into audio, and instead by using the onset times of the MIDI notes. One approach for this was attempted in an early prototype, where the IBIs for the MIDI onsets were calculated to give an estimate of the BPM, but this performed too poorly, giving wildly fluctuating BPMs and an overall inaccurate result.

Objective measure and analysis of the dynamics analysis of the system is virtually impossible, both because dynamics are so poorly defined within musical scores in the first place, and because they are interpreted by the performer. This means there is no certainty that the dynamics written in the score are what is played by the performer. However, the success of the system according to its own design can be analyzed, and the dynamic range of an audio file can be compared to the musical score for at least an approximate review of the success and limitations of the system.

First, analyzing that the dynamics are detected equally between the MIDI and audio files according to the ranges defined in Table 2. A 12 bar MIDI clip was created, with a single A4 note being played on each crotchet beat, with the velocity increasing every 2 bars. The center-points of the MIDI velocity ranges provided in Table 2 were used as the velocities for each two-bar segment. This was then synthesized using a MIDI piano sampler instrument and the MIDI and audio file were analyzed by the system. The MIDI dynamics were analyzed exactly as expected, showing a stepped climb from ***pp*** to ***ff*** over each segment. However, the detected audio dynamics which should follow this exactly do not, with the maximum detected dynamics being ***mp***. This is shown in Figure 6.

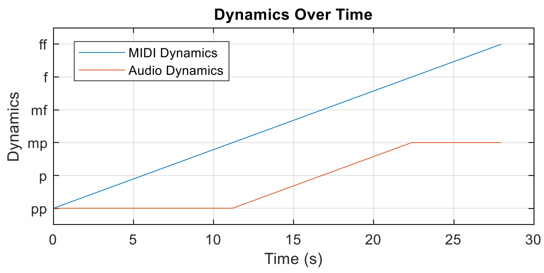


Figure 6: Graph showing MIDI and audio dynamics over time

This implies that the RMS values for analysis of the audio dynamics need adjusting in future iterations of the system, as they do not perform as expected. This can also be seen when comparing the analysis of *Gymnopedie No. 1* with the score (Satie, 2017/1888). The score shows that the piece is played quietly overall, with the most commonly occurring dynamics in the score being ***pp***. This is reflected well in the detected dynamics from the audio file, with the most common dynamic also being ***pp***. However, the score's maximum dynamic increases to ***f***, whereas the detected dynamic only increases to ***mf***, again showing that the ranges are slightly insensitive and require modification in future iterations.

As shown from the analysis, there are significant improvements which could be made to the dynamics analysis, particularly of audio signals. The first and most simple change would be to lower the RMS ranges for the louder dynamics. In an ideal case, these ranges could be determined by analyzing a large selection of piano performances with a variety of dynamics and taking the average RMS value for each different dynamic. However, this implementation for audio still suffers from relying on intensity as a measure of dynamics. Using the current implementation, it would be possible to record a soft piano performance and then increase the volume of this so that it is detected as having very loud dynamics, despite a human listener being able to easily identify the dynamics as soft. This limitation could be avoided in future iterations of the software by considering not just the intensity of the sound but also by analyzing the timbre of the piano using frequency analysis of the STFT of the signal. It is known that the strength of the first harmonic of a piano increases with pitch and that the strength of the second harmonic decreases with pitch (Repp, 1993). It has also been shown that as the dynamics of the piece get louder, the relative strength of the first harmonic increases more compared to the second, and as the dynamics get softer, the relative strength of the second harmonic decreases more compared to the first (Repp, 1993). This shows that it should be possible to use these timbral features to analyze dynamics of a piano performance, although there would undoubtedly be many challenges in detecting the correct first and second harmonics, particularly in a polyphonic performance.

4. CONCLUSION

Overall, the developed system showed mixed success in achieving its goal of providing detailed temporal and dynamics analysis of two performances and providing feedback on how the user can adjust their performance to match the professional recording more closely. The tempo detection algorithm was built upon the foundation laid by other research and performed well in tests, achieving good accuracy for both MIDI and audio signals. However, further testing should be performed to validate the success of this aspect of the system. Future developments should aim to improve the performance of this, particularly for MIDI signals. The dynamics analysis provided by the system is significantly limited, providing reasonable analysis of MIDI signals but poor analysis of audio signals. Future improvements should focus on redefining the RMS amplitude ranges for the dynamics, as well as considering timbre to avoid normalized soft performances from being detected as loud. The user interface for the system was developed to provide both detailed feedback and high-level actions to the user. This gives wide scope for the user to access detailed analysis, whilst still aiming to have a simple learning curve for less experienced users.

5. REPRODUCIBILITY

The system can be run by running the file named "MPASystem.mlapp" from the folder called "System". You will need to have MATLAB installed to run this. The audio and MIDI test files can be found in the folder named "test files" within the "system" folder and used with the system to reproduce the results from the analysis section.

6. COMPETING INTERESTS

The author has no competing interests to declare.

7. APPENDIX

A system demonstration video can be found in the supporting files, titled "DanielJacksonMPASVideo.mp4".

8. REFERENCES

Allen, P. E., & Dannenberg, R. B. (1990). Tracking musical beats in real time. In *ICMC*. www-cgi.cs.cmu.edu.

Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., & Sandler, M. B. (2005). A tutorial on onset detection in music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 13(5), 1035–1047.

Berndt, A., & Hähnel, T. (2010). Modelling musical dynamics. In *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound* (pp. 1–8). New York, NY, USA: Association for Computing Machinery.

Bowen, J. A. (1996). Tempo, duration, and flexibility: Techniques in the analysis of performance. *Journal of Musicological Research*, 16(2), 111–156.

Chafe, C., Mont-Reynaud, B., & Rush, L. (1982). Toward an Intelligent Editor of Digital Audio: Recognition of Musical Constructs. *Computer Music Journal*, 6(1), 30–41.

MIDI Manufacturers Association (1996). Channel Voice Messages. In *The Complete MIDI 1.0 Detailed Specification* (p. 42). The MIDI Manufacturers Association.

Eerola, T., & Toiviainen, P. (2016). *MIDI Toolbox 1.1: MATLAB Tools for Music Research*. Department of Music, University of Jyväskylä, Finland.

Gabrielsson, A. (2003). Music Performance Research at the Millennium. *Psychology of Music*, 31(3), 221–272. Retrieved 15 January 2021.

Gabrielsson, A. (1999) 'The Performance of Music', in D. Deutsch (ed.) The Psychology of Music, 2nd edn, pp. 501–602. San Diego: Academic Press.

Gabrielsson, A., & Juslin, P. N. (1996). Emotional Expression in Music Performance: Between the Performer's Intention and the Listener's Experience. *Psychology of Music*, 24(1), 68–91.

Intellectual Property Office. (2014, October). Exceptions to copyright: Education and Teaching.

Laroche, J. (2003). Efficient tempo and beat tracking in audio recordings. *Journal of the Audio Engineering Society*. *Audio Engineering Society*. http://www.aes.org/e-lib/browse.cfm?elib=12235.

Lerch, A. (2012). Temporal Analysis. In *An Introduction to Audio Content Analysis: Applications in Signal Processing*. Retrieved November 2020.

Percival, G., & Tzanetakis, G. (2014). Streamlined Tempo Estimation Based on Autocorrelation and Cross-correlation With Pulses. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12), 1765–1776.

Repp, B. H. (1993). Some empirical observations on sound level properties of recorded piano tones. *The Journal of the Acoustical Society of America*, 93(2), 1136–1144.

Satie, E (2017). *Gymnopédie No. 1* [Piano score]. Musescore. (Original work published 1888). https://musescore.com/classicman/scores/4766391.