Assignment GA3. Musical Key identification

Miguel Pinheiro 201705172 Diogo Remião 201706373

I. Introduction

Musical key identification is a fundamental part of musical analyses. When analysing a piece of (tonal) music, the key is a main attribute to be considered [1]. It is the basis for further music analyses, such as the mood or emotional connotation it has.

Although an important aspect to consider during the analyses process, musical key identification can only be done by well-trained people, familiar with the field of composition techniques and analyses. The common listener does not know (neither cares) in which key the song is being played. At most, the person might distinguish the key has being minor or major from the mood of the song, although completely abstracted from the theoretical implications it has [2].

Nonetheless, like any other field-of-study in STEM, different mathematical algorithms have been created with the objective of automating the process of musical key identification. We will go through them in the **Related Work** and **Method** section.

Finally, in the **Evaluation** section, we will test the implementation and go through the data we obtained, drawing the necessary conclusions.

II. Related Work

Like mentioned before, different algorithms were developed with the aim of, via a mathematical model, detect the musical key of a piece of music. However, the first aspect to take into considering is the type of analyses we are doing.

One would be to extract directly from an audio sample. This is however not the most common method has it involves creating chromagram to extract which notes are being played in each instant [3]. This is a more complex method and key detection from musical audio solutions are still limited. Furthermore, in the evaluation section, the precision of the extraction of pitches would also have to be considered, adding another complexity layer to the problem.

We therefore opted for extracting the data from MIDI files, where we make use of symbolic data, and therefore obtain ground-true information related to the piece of music [4]. This will further considered in the **Implementation** section.

Having extracted the data, the next step is to apply and algorithm that outputs a prediction. The algorithm we will be making use of is the **Maximum Key-profile Correlation** (MKC) [5] based on key profiles. This algorithm will be further analysed in the **Method** section.

A number of studies have made use of this algorithm, or created variations of it, to compute the musical key. [6] proposed revised versions of the key profiles initially presented by Krumhansl [5] in order to improve the accuracy of the algorithm. [7] identifies a possible inaccuracy of the algorithm related to the window size choice, offering as a solution applying filters to smooth out local oscillations and impulses that might affect the accuracy of the algorithm.

Some studies have also used this algorithm for other purposes, like in [8] where MKC is used to measure the tonality, colloquially know as ”sounding nice”, of a piece of music. [9] made use of MKC for chord recognition. This is also based on profiles, with one for each chord just like we would have one for each key.

There are alternatives to the Krumhansl-Schmukler algorithm, like in [10], where the author makes use of a geometrical representation called the Spiral Array, in which pitches are represented by points on a spiral and determines key with a Center of Effect Generator method.

III. Method

In this section we will go through the algorithm used to determine the musical key. As stated before, we will make use the MKC proposed by Krumhansl in [5].

The Krumhansl-Schmuckler algorithm is based on key profiles. Each one of the 24 possible keys (A-G#, Major and Minor) will have its own profile. Each profile is a vector of 12 values, each of them corresponding to a note in the chromatic scale. (A-G#). The profile was created based on data from experiments by Krumhansl and Kessler [11]. In these experiments, participants where asked to rate how well a note fitted a musical element like a scale, chord and cadence, associated with a key. We these ratings, a weighted value was given to each note. The experiment was repeated for all keys.

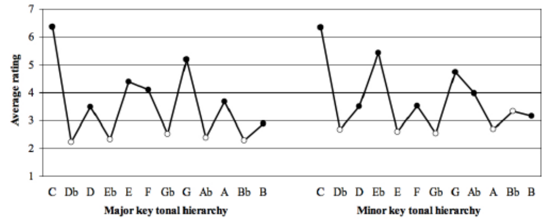

Figure 1. Key profiles for C Major and C Minor by Krumhansl & Kessler's (1982)

As we can observe in figure 1, in both profiles, the **tonic** note (C) is given the highest weight. This is because it is the most important note, with music being composed around it. It is therefore, the outcome that our ear is expecting naturally for example, in a cadence.

Another highly important note is the **dominant** note, in this case G for both of them. The dominant is the note that effectively enables us to identify the key of the piece(A-G#). It it part of the tonic chord, and when we hear the dominant note/chord, we expected it to lead to the tonic note/chord, which we call the perfect cadence.

Finally, we can also pinpoint the **mediant** note. The mediant is the note that differentiates a major and minor key in the same note. In the case, the mediant is E and Eb for C Major and C Minor respectively. This note is also part of the tonic chord, and theoretically, it can render the dominant note useless as it also identifies the key. Switching the dominant for the mediant is, in fact, a very common composition technique in Jazz. It tends to have a more significant role in a a minor key as, with only the tonic and dominant notes, we tend to assume the key is major. Therefore, its recursive use is necessary to define the key as minor.

Went the data was collected for all keys, it was noticed that there was little variation between major keys (after adjusting for transposition). For example, in C major, C weighted 6.35 and C# 2.23, whereas in C# major, C# weighted 6.35 and D 2.23. For this matter, the values were averaged over all major keys, creating a profile that was used in all major keys. The same was done for minor keys.

In order to compare the piece with the different profiles, we to create a vector of 12 inputs. Each input corresponds to the total duration of the pitch (A-G#) in the piece. Of course, the values obtained depend on the tempo at which the piece is being played at. However, every value will scale proportionately with the tempo, therefore affecting the output value but not the output prediction.

The correlation between the input vector and each profile is given by:

$$r = \frac{\sum (x-\overline{x})(y-\overline{y})}{(\sum (x-\overline{x})^2(y-\overline{y})^2)^{1/2}}$$

where $x =$ is the input vector values, $\overline{x} =$ the average input vector values, $y =$ the key-profile values for a given key and $\overline{y} =$ the average key-profile values for that key.

The calculation must be done for all key profiles, and the one that yields the highest value is the preferred key.

In [6], Temperley calculated the input vector values for a small melody called "Yankee Doodle":

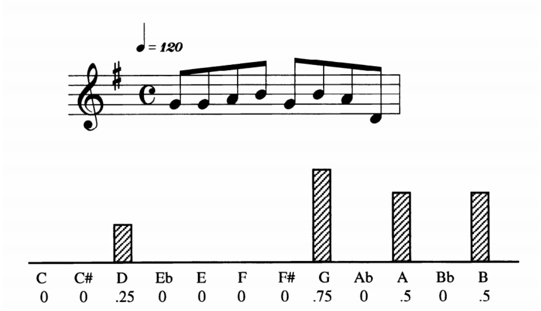

Figure 2. Measure of "Yankee Doodle," with input vector showing total duration of each pitch class

The correlation values obtained for each key profile are the following:

Key Profile Scores

Key	Score
C major	0.245
C minor	-0.012
C# major	-0.497
C# minor	-0.296
D major	0.485
D minor	0.133
D# major	-0.114
D# minor	-0.354
E major	0.000
E minor	0.398
F major	0.003
F minor	-0.384
F# major	-0.339
F# minor	0.010
G major	**0.693**
G minor	0.394
G# major	-0.432
G# minor	-0.094
A major	0.159
A minor	0.223
A# major	-0.129
A# minor	-0.457
B major	-0.061
B minor	-0.436

Table I

As it is observed in table I, **G major** is the preferred key. In fact, it is the correct answer. This is a relative simple example, as all the notes in the excerpt are in the G major scale. Also, the tonic and mediant notes have a significant presence, and its dominant is also present, making it very obvious for the algorithm.

IV. Implementation

Having described the algorithm, the next step is to implement it programmatically. This is because if we want to obtain a significant number of results, doing calculations by hand would be impractical.

The implementation was done in Python, a high level programming language, with lots of libraries and tools for multimedia projects. We made use of the library *Music21*, a Python-based toolkit for computer-aided musicology [12].

This library has a key determination algorithm implemented based on the *Krumhansl-Schmuckler* key determination algorithm. This is the one we will be making use in this project.

One advantage of this library is that it allows to specify which key-profiles to use when calculation the correlation with the sample. This will allow us to compare different key profiles and their respective accuracy relatively easily. For the purpose of the project, we will be explaining in relative detail how to use this library, detailing some classes and functions crucial in this project.

A. *music21.stream*

The class *music21.stream* is the fundamental container in *music21*. It is in this class where objects may be ordered and/or placed in time based on offsets from the start of this container. This means that it basically encodes music elements on a time-based system, effectively creating a file system that the rest of the library can understand.

Although it is possible to ”manually” create a *Stream*, defining which notes are to be played, as well as their time-frame, this is not necessary nor recommended. What we will be doing in converting a music file into a *Stream* using the function *music21.converter.parse*.

This function accepts a number of symbolic music files, files where the data is structured, and therefore well defined if we know how to read it. A few examples can be named like MusicXML, MuseData, Humdrum and most importantly, MIDI.

What is consistent amongst all these file types is that, although they store musical data, **none of them are in fact audio files**. Although audio files, such as MP3 and WAV, enables us to play music in virtually any device, it does not contain any information regarding what is actually being played, i.e, we do not know whether a piano or a flute is playing simply by reading the file. This makes it very difficult to extract information like which notes are being played, which is the essence of key determination for example.

For this matter, we use file types like MIDI, where all this information is available directly, simplifying significantly the process. By parsing a MIDI file into a *Stream*, we are know able to make use of the library’s analyses function. It is important to note that some equalization in applied in order to improve the algorithm’s efficiency. However, most excerpts will be unaffected as this equalization is done using a very high tempo subdivision.

B. *music21.analysis.discrete.KeyWeightKeyAnalysis*

Once we have created a **Stream** based on a MIDI file, we can now apply analysis tools to obtain different parameters. One of such tools is the *KeyWeightKeyAnalysis*, the implementation of the key determination algorithm.

To be precise, this class does not exist, i.e, you cannot define an object using this class. It is what is called a base class. This means that for example classes *...discrete.a* and *...discrete.b*, both subclasses of *...discrete.KeyWeightKeyAnalysis*, inherit the same base functions, although they can have specific functions for them.

The subclasses in this case will different key-profiles alternatives that can be applied in the algorithm:

- SimpleWeights - Implementation of simple weights by Craig Sapp [13]
- AardenEssen - Implementation of Aarden-Essen weightings [14]
- BellmanBudge - Implementation of Bellman-Budge weightings [15]
- KrumhanslSchmuckler - Implementation of Krumhansl-Schmuckler/Kessler weightings [5]
- TemperleyKostkaPayne - Implementation of Temperley-Kostka-Payne weightings [6]

Like mentioned before, this subclasses output a different correlation value as they make use of different key profiles.

Although not necessary, it is possible to see which profiles are being use for both major and minor keys in each implementation, by making use of the function *KeyWeightKeyAnalysis.getWeights(weightType)*. For example, this is the output of the Krumhansl-Schmuckler/Kessler weightings, represented in Figure 1:

```
Major Key profile [6.35, 2.23, 3.48, 2.33, 4.38, ↩
    4.09, 2.52, 5.19, 2.39, 3.66, 2.29, 2.88]
Minor Key profile [6.33, 2.68, 3.52, 5.38, 2.6, 3.53,↩
     2.54, 4.75, 3.98, 2.69, 3.34, 3.17]
```

C. *music21.stream.analyze*

Although we can do the analysis by calling IV-B, it is easier to simply call *stream.analyze(arg)*. This function runs a particular analytical method on the contents of the specified stream to find its key in this case, essentially calling *music21.analysis.discrete.KeyWeightKeyAnalysis* functions. For example, *stream.analyze(’key’)* would output the prediction based on the implementation of Krumhansl-Schmuckler/Kessler weightings, which is the default one. Making use of the subclass *music21.analysis.discrete.analyzeStream*, we can specify which implementation we want to use. This subclass matches the argument string with the implementations available:

- analysis.discrete.analyzeStream(s, ’Krumhansl’) = stream.analyze(’Krumhansl’) - Outputs prediction based on Krumhansl profiles
- analysis.discrete.analyzeStream(s, ’Temperley’) = stream.analyze(’Temperley’) - Outputs prediction based on Temperley profiles

D. *music21.key.Key*

Independently of the method used to make the prediction the output will be an object of the class *music21.key.Key*. As described in the documentation, ”Note that a key is a sort of hypothetical/conceptual object. It probably has a scale (or scales) associated with it and a KeySignature, but not necessarily”.

The **key signature** is the modification associated with a particular key in comparison to C Major. It is the number of sharps or flats it has. For example, G Major’s key signature has two sharps (F# and C#), while C Major has none.

A **scale** in the sequence of notes ordered by pitch starting from the key’s tonic tonic and ending in the same (octave), taking into account the key signature of said key. For example, G Major’s scale is ”G A B C# D E F# G”, while C Major’s is ”C D E F G A B C”.

These concepts are usually associated with tonal music, hence the small disclaimer in the documentation. As only tonal music will be analysed, both parameters will be defined.

Although the above described functions will already output a **music21.key.Key** object, one can be easily defined:

```
>> cm = key.Key('c')  # lowercase = c minor.
>> cm
<music21.key.Key of c minor>
>>cm.mode
 'minor'
>>cm.tonic.name
 'C'
>>cm.sharps
-2
>>cm.pitches
[<music21.pitch.Pitch C4>,
<music21.pitch.Pitch D4>,
<music21.pitch.Pitch E-4>,
<music21.pitch.Pitch F4>,
<music21.pitch.Pitch G4>,
<music21.pitch.Pitch A-5>,
<music21.pitch.Pitch B5>,
<music21.pitch.Pitch C5>]
```

As observed, when we define a C Minor key, the object will have an associated key signature, as well as a scale. All the returned objects form the analyses will be defined like this one, allowing us to make use of a large set of functions associated with this object.

The most important ones are arguably *key.tonic.name* and *key.mode*, which output the **pitch** (A, B, C, etc) and the **mode** (Major or Minor). These are the two main parameters that will be used to analyse the precision of the algorithm. In short, **these define the key of the excerpt**.

Given that these parameters are output as *strings*, a comparison can easily be made with the name of the file, which will contain the key itself (**Testing** section).

A number of other functions are also useful for this project. *key.relative* in a key object containing the relative of the main key. The **relative** in key that has the shares the same key signature as the main key, except the leading tone. For example, the relative of G Major is E Minor, whose only difference is the D#, the leading tone, which needs to be half-tone higher in minor keys.

Similarly, *key.getDominant()* returns the **Dominant** pitch of the main key. Although the Dominant does not share the same key signature, its tonic chord is a recurring presence in the main key, having therefore big similarities. It is important to note that the object returned by this function is not *music21.key.Key* but rather *music21.pitch.Pitch*. Although its brings different considerations into play, the dominant chord, and by extrapolation the key, is always major. Therefore, we only need to know the pitch, which is similarly obtained using *.name*.

Finally, *key.parallel* returns a key object containing the **parallel** key of the main key. In short, its is the same pitch with opposite mode (A Major to A Minor). The only thing these two keys share it a common tonic note, which can be enough to confuse the algorithm into prediction the parallel instead ot the main key.

The above described functions are the essential ones for this project, giving us all the information we need to make the necessary comparisons. However, it is possible to go further into detail, in order to analyse the ambiguity of a prediction, for example. In other words, how sure was the algorithm about its prediction: by a landslide, or was it a close match? This information in related to the correlation value associated with each key.

This can be obtained using *key.correlationCoefficient* for the chosen key. If we want to know the results for the rest of the profiles, *key.alternateInterpretations[n]* is a vector with the (n+2)-est best prediction. Each of this entries is of course, a key object, so we can apply all the above functions on them as well:

```
Best Prediction =  A major 0.8769757225959743
Second best Prediction =  G- minor 0.7810592206505336
Third best Prediction =  E major 0.7077655748628783
Worst Prediction =  E- major -0.7693194496397553
```

The above textbox shows the output for *Prelude in A Major, J. S. Bach* using the Krumhansl key profile. Not only do we see which one was the chosen key, but also the rest of podium, as well as the respective correlation values.

V. Evaluation

A. Data set

The first aspect to mention is the data set used. In order to diversify the test, different sample groups were created and analysed separately. This is because, if we are to make a fair judgement on the algorithms precision, we need to understand the theoretical implications each music sample has.

The first sample group is **J.S.Bach's Well Tempered Clavier**. This is a compilation of 24 pieces written for each of the possible keys. For this matter, it is an excellent data set as it goes through all possible solutions.

The second sample group in **F.Chopin's Études**. Similarly, these 18 pieces have very diversified keys, presenting a good testing opportunity of the algorithm. However, it is important to mention that Chopin and Bach have very different composition techniques. This means that both data sets will bring different considerations.

The third sample group is **S.Rachmaninov's Piano Concertos 2 & 3**. Although this data set only has 6 samples (3 for each *concerto*), they are much longer than previous ones. This means that each sample is much more diverse and has more variations , making it much harder for the algorithm to predict the key.

The forth sample group is **Contemporary Music**, with 7 pieces including Debussy, Ravel and Satie. This composers had a different understating of tonality, almost disregarding it. In other words, given that the models were based on tonal perception, we expect that the algorithm will not do so well in this data set.

The last sample group is **Pop Music**, with 8 pieces. This genre differs from the rest of the samples, as it is simpler and repetitive, even amongst themselves. In fact, the chord progressing is most of the time very simple, consisting on perfect cadences that give away the key instantly. Therefore, it is expected that the algorithm will do very well it this case.

B. Analyses Considerations

As mentioned before, although we will make use of the Krumhansl algorithm, different key can be used. We will test the precision of some of them, which are implemented in library we are using.

The different key profiles we will be testing are the ones mentioned in section IV-B.

Not only will we evaluate the precision of the algorithm, i.e., if it predicted the right answer, but also analyse the wrong answers. As mentioned in IV-D, some key profiles are very similar to each other, namely:

- Relative
- Dominant
- Parallel

Although still wrong answer, they are understandable mistakes because of the similarity, and analysing the excerpt might gives us some insights on the algorithm's output.

C. Bach Analyses

%	Krum	Temp	Bell	Aard	Craig
Correct	100	100	100	100	100
Relative	0	0	0	0	0
Dominant	0	0	0	0	0
Parallel	0	0	0	0	0
Other	0	0	0	0	0

As we can observe from the table above, all the algorithms have 100% accuracy. This is more-less to be expected. The composition style in this pieces is very uniform. Perfect cadences are executed multiple times during the piece. Perfect cadences correspond to the chord progression I - IV - V -I, which have all the major pitches in that key. This way, because this pitches have a much higher weight in the analyses it will be easier to identify the key.

Another aspect to take consideration is modulations. Modulations consists in changing the main key of a section of the piece. Bach often makes use of a modulation technique called *Circle of fifths*. This technique enables to modulation to many keys seamlessly to closely related keys. These keys share many notes, and therefore do not impact much the prediction unless they are used very often.

Preludes should also be easier to analyse than Fugues, and in this case they were analysed together.

D. Chopin Analyses

%	Krum	Temp	Bell	Aard	Craig
Correct	94.44	94.44	94.44	94.44	100
Relative	0	5.56	5.56	0	0
Dominant	5.56	0	0	0	0
Parallel	0	0	0	0	0
Other	0	0	0	5.56	0

Although from a different age than Bach (Baroque), Chopin (Romantic) composed based on tonal rules and very fluently melodies and chords. These meant that modulating were once again often used with closely related keys, like Dominant, sub-dominant and relative keys. In these pieces in particular, modulating sections were long.

This explains while we see some predictions falling in other categories (the ”other” prediction in ”Aarden” was in fact to the sub-dominant). It also needs to be taken into account that *Études* are composed to train technique, often being simpler in terms of melody and harmonization. These factors of course make it easier for the algorithm to work.

E. Rachmaninov Analyses

%	Krum	Temp	Bell	Aard	Craig
Correct	16.67	33.33	33.33	33.33	33.33
Relative	0	0	0	0	0
Dominant	0	0	0	16.67	0
Parallel	16.67	0	0	0	0
Other	66.67	66.67	66.67	50	66.67

Rachmaninov piano concertos are the climax of the romantic period. A period defined by emotions and a megalomaniacal Russian culture result in very intense and diverse music, with these piano concertos being arguably the best representation of that. Being such long pieces, we can identify different parts using different keys, techniques and tempos. We can define these pieces as literally ”All over the place” in terms on music composition.

Therefore, it is understandable the difficulty the algorithms have in correctly predicting the answer. This category was in fact selected precisely to prove the major flaw in Krumhansl approach: no segmentation of the pieces. When you have a long piece that modulates into different keys for long periods of time, it is impossible to predict the main key using this algorithm. In fact it does not even make any sense to analyse these pieces in such a generic way, as their complexity calls for a more local approach.

F. Contemporary Analyses

%	Krum	Temp	Bell	Aard	Craig
Correct	57.14	57.14	57.14	42.86	57.14
Relative	0	0	0	14.29	0
Dominant	14.29	14.29	14.29	14.29	14.29
Parallel	0	0	0	0	0
Other	28.57	28.57	28.57	14.29	28.57

Contemporary music stands out for not following formal tonal music rules. Although the pieces chosen are still from the early part of the movement, we can already identify, for example, that modulations do not make use of any technique in particular. The composer bases its composition in the sonic effect rather than the formal modulation techniques. For example, cadences are rarely perfect. In fact, tonal chords progressions are very rare.

As we are applying a tonal algorithm to ”border-tonal” music, we can accept the 50% accuracy obtained as a good performance from the algorithms. Especially considering that most mistakes were understandable, predicting the dominant of relative keys which, as explained in section IV-D, have very similar profiles.

G. *Pop Analyses*

%	Krum	Temp	Bell	Aard	Craig
Correct	75	75	100	100	87.5
Relative	0	0	0	0	0
Dominant	25	0	0	0	0
Parallel	0	0	0	0	0
Other	0	25	0	0	12.5

Pop music, as mentioned before, is very simple and makes use of repetitive chord progressions. Modulations are rarely made, and when they are, it is using closely related keys. Because these keys all share most pitches in common, some mistakes are understandable. However some algorithms got 100% accuracy, proving that in fact, this is the easiest genre to analyse. Bach might seem to be better, but Bach's sample list was bigger than Pop's. Therefore a direct comparison is not possible. As argument can be made given the fact that analysing by hand a piece of Bach might take a day, while Pop music takes 1 minute.

H. *Profile Comparison*

%	Krum	Temp	Bell	Aard	Craig
Correct	68.65	71.98	76.98	74.13	75.59
Relative	0	1.11	1.11	2.86	0
Dominant	8.96	2.86	2.86	6.19	2.86
Parallel	3.33	0	0	0	0
Other	19.06	24.05	19.05	16.82	21.55

Comparing the average performance of the different algorithms, the original algorithm, Krumhansl, has the worst performance, while Bellman's key profiles have the best results by a significant margin.

The original key profiles were based completely in a subjective analyses done with volunteers based on ”sounding nice”. Although this might work most of the time, it only goes so far, as complex musical properties come into play when the excerpts are more complex. It is expected that, with more technology and more research, newer models were suggested after studying the original model and identifying its flaws.

Another aspect to take into consideration is the type of weights given to each key. In Krumhansl, the tonic has a weight of 6.35, with the next pitch (not part of the scale of that key) having a weight of 2.23. In Aarden, the tonic weights 17.7661, with next pitch weighting 0.145624.

We can see that in Aarden / Bellman profiles, much bigger importance is given to the notes that are part of the scale of that pitch, in contrast with Krumhansl / Temperley. This reduces the error caused by ornamentations of passing notes. Simple (Craig) profiles give 0 weight to pitches that are not part of the scale and 1 to the ones that are, with the tonic weighting 2.

Given that the sample list is not that extended and/or diverse, the algorithms can be said to be performing at more less the same level, with the revisions slightly outperforming in some more complex categories.

VI. Conclusion

We start by extracting musical information from the MIDI files and compile it into the libraries *music21.stream* class. After this conversion is done, a distribution profile is created with each pitch in the chromatic scale total usage time. The Krumhansl algorithm is then applied, where a correlation between previously defined key-profiles for each possible key and the sample are done. The highest output is then presented as the determined key.

Five key-profiles proposals were tested, with Bellman-Budge key profiles having the best accuracy, with 76.98% correct answers The oldest and original key profiles proposed by krumhansl-Schmuckler/Kessler had the worst precision of all, with 68.65% correct answers.

In terms of neighbouring keys, Temperley-Kostka-Payne had the worst performance, with 24.05% of the answers not falling in neither the tonic, relative, dominant of parallel keys. On the other hand, Aarden-Essen key-weightings presented only 16.82% of answers not falling in neither of the above categories.

In general, the correlation method presented by krumhansl had an average accuracy of 73.466%.

As far as the comparison with human performance is concerned, with do not have the desired data that to make that analyses. One study suggested that musically trained people can identify the correct key in 75% of the cases, tough after listening only to the first measure [16]. This experiment was also conducted using samples from *J.S.Bach Well Tempered Clavier*, in which algorithms in this paper had 100% accuracy but could analyse both the *Prelude & Fugue* to the full extent.

It is also important to know that in order to compare the types of mistakes and approaches from humans and algorithms, further research would be required.

Concluding, we indicate the following weak points in this project:

- Only the Krumhansl correlation algorithm was tested, with other approaches not taken into consideration
- The number of samples and its diversity is not enough to do a fair comparison between the different key-weightings. More samples would be required to obtain accuracy levels closer to the actual algorithms performance levels.
- The testing was conducted using all the time frame of the pieces. These pieces have different parts that use different keys, so such a simple approach is not fair to evaluate the accuracy of the algorithms. *Rachmaninov's concertos* are a good example of this flaw.

References

[1] D. Temperley, *The cognition of basic musical structures*. MIT press, 2004.

[2] R. Parncutt, “The emotional connotations of major versus minor tonality: One or more origins?” *Musicae Scientiae*, vol. 18, no. 3, pp. 324–353, 2014.

[3] S. Pauws, “Musical key extraction from audio.,” in *ISMIR*, 2004.

[4] C. Raffel and D. Ellis, “Extracting ground-truth information from midi files: A midifesto,” in *ISMIR*, 2016.

[5] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford University Press, 2001.

[6] D. Temperley, "What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered," in *Music Perception: An Interdisciplinary Journal*. University of California, 1999, pp. 65–100.

[7] O. Y. Harja and I. Shmulevich, "Graph based smoothing of class data with applications in musical key finding,"

[8] A. H. Takeuchi, "Maximum key-profile correlation (mkc) as a measure of tonal structure in music," *Perception & Psychophysics*, vol. 56, no. 3, pp. 335–346, 1994.

[9] K. Lee, "Automatic chord recognition from audio using enhanced pitch class profile," in *ICMC*, Citeseer, 2006.

[10] E. Chew, "Towards a mathematical model of tonality," Ph.D. dissertation, Massachusetts Institute of Technology, 2000.

[11] C. L. Krumhansl and E. J. Kessler, "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys.," *Psychological review*, vol. 89, no. 4, p. 334, 1982.

[12] M. S. Cuthbert, *Music21 – a toolkit for computational musicology*, https://github.com/cuthbertLab/music21, 2021.

[13] C. S. Sapp, *Computational methods for the analysis of musical structure*. Stanford University, 2011.

[14] B. J. Aarden, "Dynamic melodic expectancy," Ph.D. dissertation, The Ohio State University, 2003.

[15] H. Bellmann, "About the determination of key of a musical excerpt," in *International Symposium on Computer Music Modeling and Retrieval*, Springer, 2005, pp. 76–91.

[16] A. J. Cohen, "Tonality and perception: Musical scales prompted by excerpts from das wohl-temperierte clavier of js bach," in *Second Workshop on Physical and Neuropsychological Foundations of Music, Ossiach, Austria*, 1977.