Tarsos, a Modular Platform for precise Pitch Analysis of Western and non-Western music

Joren Six[1], Olmo Cornelis[1], and Marc Leman[2]

[1]University College Ghent, School of Arts
Hoogpoort 64, 9000 Ghent - Belgium
joren@0110.be
olmo.cornelis@hogent.be

[2]Ghent University, IPEM - Department of Musicology
Blandijnberg 2, 9000 Ghent - Belgium
marc.leman@ugent.be

August 22, 2013

Abstract

This paper presents Tarsos, a modular software platform used to extract and analyze pitch organization in music. With Tarsos pitch estimations are generated from an audio signal and those estimations are processed in order to form musicologically meaningful representations. Tarsos aims to offer a flexible system for pitch analysis through the combination of an interactive user interface, several pitch estimation algorithms, filtering options, immediate auditory feedback and data output modalities for every step. To study the most frequently used pitches, a fine-grained histogram that allows up to 1200 values per octave is constructed. This allows Tarsos to analyze deviations in Western music, or to analyze specific tone scales that differ from the 12 tone equal temperament, common in many non-Western musics. Tarsos has a graphical user interface or can be launched using an API - as a batch script. Therefore, it is fit for both the analysis of individual songs and the analysis of large music corpora. The interface allows several visual representations, and can indicate the scale of the piece under analysis. The extracted scale can be used immediately to tune a MIDI keyboard that can be played in the discovered scale. These features make Tarsos an interesting tool that can be used for musicological analysis, teaching and even artistic productions.

Keywords: Pitch Detection, Computational Ethnomusicology, Pitch Class Histogram, Tone Scale Extraction

1 Introduction

In the past decennium, several computational tools became available for extracting pitch from audio recordings (Clarisse et al., 2002; Cheveigné & Hideki, 2002; Klapuri, 2003). Pitch extraction tools are prominently used in a wide range of studies that deal with analysis, perception and retrieval of music. However, up to recently, less attention has been paid to tools that deal with distributions of pitch in music.

The present paper presents a tool, called Tarsos, that integrates existing pitch extraction tools in a platform that allows the analysis of pitch distributions. Such pitch distributions contain a lot of information, and can be linked to tunings, scales, and other properties of musical performance. The tuning is typically reflected in the distance between pitch classes. Properties of musical performance may relate to pitch drift within a single piece, or to influence of enculturation (as it is the case in African music culture, see Moelants et al. (2009)). A major feature of Tarsos is concerned with processing audio-extracted pitches into pitch and pitch class distributions from which further properties can be derived.

Tarsos provides a modular platform used for pitch analysis - based on pitch extraction from audio and pitch distribution analysis - with a flexibility that includes:

- The possibility to focus on a part of a song by selecting graphically displayed pitch estimations in the melograph.
- A zoom function that allows focusing on global or detailed properties of the pitch distribution.
- Real-time auditory feedback. A tuned MIDI synthesizer can be used to hear pitch intervals.
- Several filtering options to get clearer pitch distributions or a more discretized melograph, which helps during transcription.

In addition, a change in one of the user interface elements is immediately propgated through the whole processing chain, so that pitch analysis becomes easy, adjustable and verifiable.

This paper is structured as follows. First, we present a general overview of the different processing stages of Tarsos, beginning with the low level audio signal stage and ending with pitch distributions and their musicological meaning. In the next part, we focus on some case studies and give a scripting example. The next part elaborates on the musical aspects of Tarsos and refers to future work. The fifth and final part of the main text contains a conclusion.

2 The Tarsos Platform

Figure 1 shows the general flow of information within Tarsos. It starts with an audio file as input. The selection of a pitch estimation algorithm leads to a pitch estmations, which can be represented in different ways. This representation can be further optimized, using different types of filters for peak selection. Finally, it is possible to produce an audio output of the obtained results. Based on that output, the analysis-representation-optimization cycle can be refined. All steps contain data that can be exported in different formats. The obtained pitch distribution and scale itself can be saved as a scala file which in turn can be used as input, overlaying the estimation of another audio file for comparison.

In what follows, we go deeper into the several processing aspects, dependencies, and particularities. In this section we first discuss how to extract pitch estimations from audio. We illustrate how these pitch estimations are visualized within Tarsos. The graphical user interface is discussed. The real-time and output capabilities are described, and this section ends with an explanation about scripting for the Tarsos API. As a reminder: there is a manual available for Tarsos at `http://tarsos.0110.be/tag/JNMR`.

2.1 Extracting pitch estimations from audio

Prior to the step of pitch estimation, one should take into consideration that in certain cases, audio preprocessing can improve the subsequent analysis within Tarsos. Depending on the source material and on the research question, preprocessing steps could include noise reduction, band-pass filtering, or harmonic/percussive separation Nobutaka et al. (2010). Audio preprocessing should be done outside of the Tarsos tool. The, optionally preprocessed, audio is then fed into Tarsos and converted to a standardized format[1].

The next step is to generate pitch estimations.

[1] The conversion is done using FFMPEG, a cross platform command line tool to convert multimedia files between formats. The default format is PCM WAV with 16 bits per sample, signed, little endian, 44.1kHz. Furthermore, all channels are down mixed to mono.

This is an Author's Original Manuscript of an article whose final and definitive form, the Version of Record, has been published in the Journal of New Music Research (JNMR), August 2013, © Taylor & Francis.
Available online at `http://www.tandfonline.com/10.1080/09298215.2013.797999` .

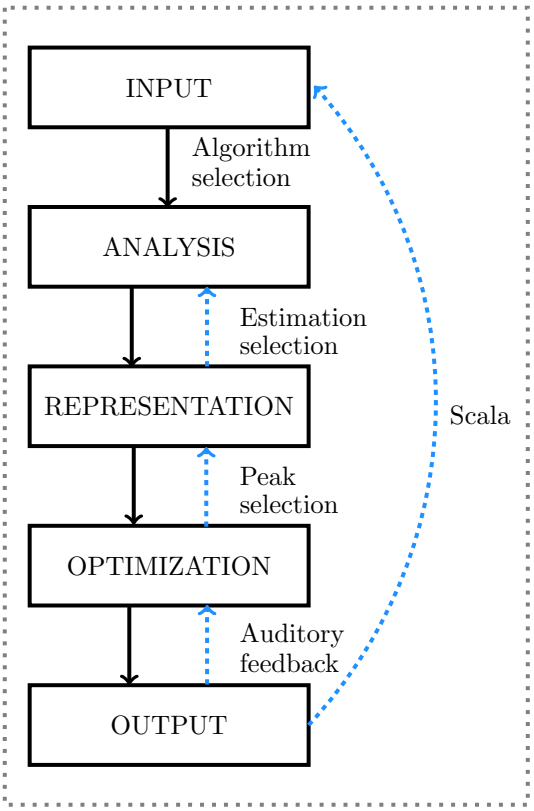


Figure 1: The main flow of information within Tarsos.

Each selected block of audio file is examined and pitches are extracted from it. In figure 2, this step is located between the input and the signal block phases. Tarsos can be used with external and internal pitch estimators. Currently, there is support for the polyphonic MAMI pitch estimator (Clarisse et al., 2002) and any VAMP plug-in (Cannam et al., 2010) that generates pitch estimations. The external pitch estimators are platform dependent and some configuration needs to be done to get them working. For practical purposes, platform independent implementations of two pitch detection algorithms are included, namely, YIN (Cheveigné & Hideki, 2002) and MPM (McLeod & Wyvill, 2005). They are available without any configuration. Thanks to a modular design, internal and external pitch detectors can be easily added. Once correctly configured, the use of these pitch modules is completely transparent, as extracted pitch estimations are transformed to a unified format, cached, and then used for further analysis at the symbolic level.

2.2 Visualizations of pitch estimations

Once the pitch detection has been performed, pitch estimations are available for further study. Several types of visualizations can be created, which lead, step by step, from pitch estimations to pitch distribution and scale representation. In all these graphs the *cent* unit is used. The cent divides each octave into 1200 equal parts. In order to use the cent unit for determining absolute pitch, a reference frequency of 8.176Hz has been defined[2], which means that 8.176Hz equals 0 cents, 16.352Hz equals 1200 cents and so on.

A first type of visualization is the melograph representation, which is shown in Figure 3. In this representation, each estimated pitch is plotted over time. As can be observed, the pitches are not uniformly distributed over the pitch space, and form a clustering around 5883 cents.

A second type of visualization is the pitch histogram, which shows the pitch distribution regardless of time. The pitch histogram is constructed by

[2]See Appendix B for a discussion about pitch representation in cents and the seemingly arbitrary reference frequency of 8.176Hz.

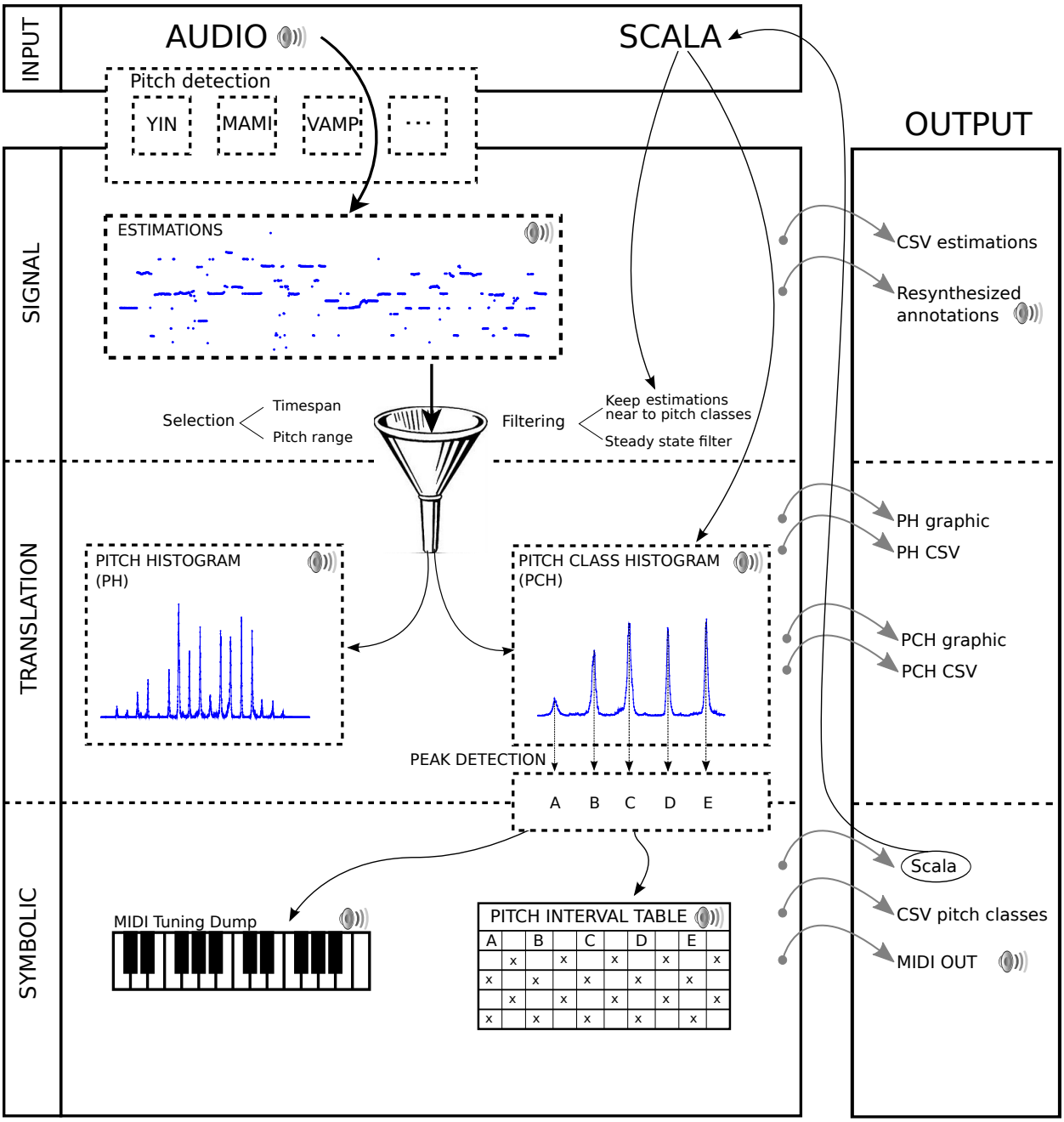


Figure 2: Detailed block diagram representing all components of Tarsos, from input to output, from signal level to symbolic level. All additional features (selection, filtering, listening) are visualized (where they come into play). Each step is described into more detail in Chapter 3.

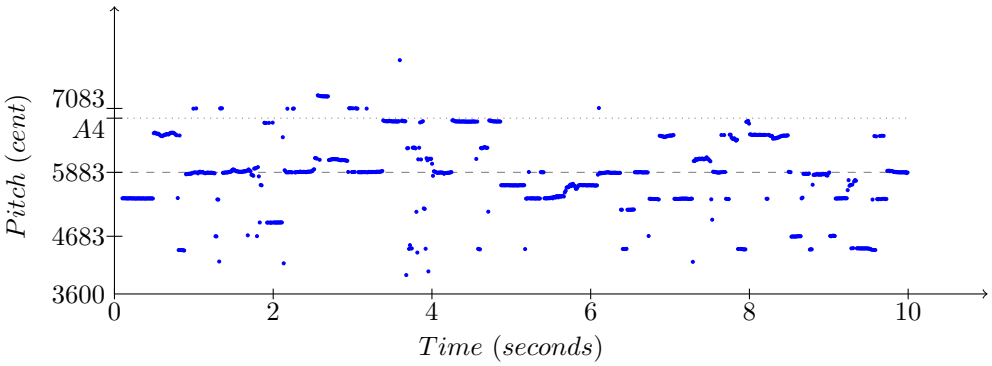


Figure 3: A melograph representation. Estimations of the first ten seconds of an Indonesian Slendro piece are shown. It is clear that pitch information is horizontally clustered, e.g. the cluster around 5883 cents, indicated by the dashed horizontal line. For reference a dotted horizontal line with $A4$, 440Hz is also present.

assigning each pitch estimation in time to a bin between 0 and 14400[3] cents, spanning 12 octaves. As shown in Figure 4, the peak at 5883 cents is now clearly visible. The height of a peak represents the total number of times a particular pitch is estimated in the selected audio. The pitch range is the difference between the highest and lowest pitch. The graph further reveals that some peaks appear every 1200 cents, or every octave.

A third type of visualization is the pitch *class* histogram, which is obtained by adding each bin from the pitch histogram to a corresponding modulo 1200 bin. Such a histogram reduces the pitch distribution to one single octave. A peak thus represents the total duration of a pitch *class* in a selected block of audio. Notice that the peak at 5883 cents in the pitch histogram (Figure 4) now corresponds to the peak at 1083 cents in the pitch class histogram (Figure 6).

It can also be useful to select only filter pitch estimations that make up the pitch class histogram. The most obvious 'filter' is to select only an interesting timespan and pitch range. The distributions can be further manipulated using other filters and peak detection. The following three filters are implemented in Tarsos:

The first is an *estimation quality* filter. It simply

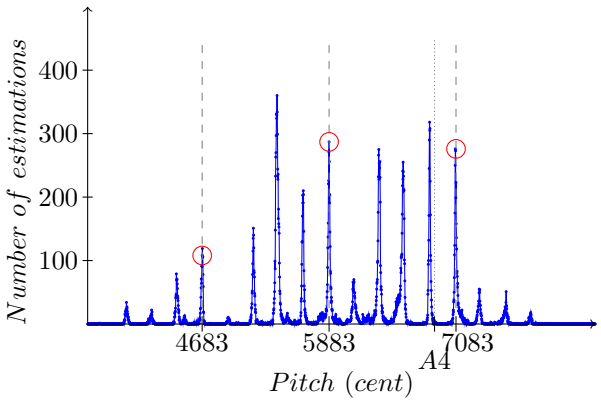


Figure 4: A pitch histogram with an Indonesian Slendro scale. The circles mark the most estimated pitch classes. The dashed vertical lines show the same pitch class in different octaves. A dotted vertical line with $A4$, 440Hz, is used as a reference for the diapason.

[3] 14400 absolute cents is equal to 33488Hz, well above human hearing.

removes pitch estimations from the distribution below a certain quality threshold. Using YIN, the quality of an estimation is related to the periodicity of the block of sound analyzed. Keeping only high quality estimations should yield clearer pitch distributions.

The second is called a *near to pitch class filter*. This filter only allows pitch estimations which are close to previously identified pitch classes. The pitch range parameter (in cents) defines how much ornamentations can deviate from the pitch classes. Depending on the music and the research question, one needs to be careful with this - and other - filters. For example, a vibrato makes pitch go up and down - pitch modulation - and is centered around a pitch class. Figure 5a gives an example of Western vibrato singing. The melograph reveals the ornamental singing style, based on two distinct pitch classes. The two pitch classes are hard to identify with the histogram 5c but are perceptually there, they are made clear with the dotted gray line. In contrast, figure 5b depicts a more continuous glissando which is used as a building block to construct a melody in an Indian raga. For these cases, Krishnaswamy (2004b) introduced the concept of two-dimensional 'melodic atoms'. (In Henbing & Leman (2007) it is shown how elementary bodily gestures are related to pitch and pitch gestures.) The histogram of the pitch gesture Figure 5d suggests one pitch class while a fundamentally different concept of tone is used. Applying the near to pitch class filter on this type of music could result into incorrect results. The goal of this filter is to get a clearer view on the melodic contour by removing pitches between pitch classes, and to get a clearer pitch class histogram.

The third filter is a *steady state filter*. The steady state filter has a time and pitch range parameter. The filter keeps only consecutive estimations that stay within a pitch range for a defined number of milliseconds. The default values are 100ms within a range of 15 cents. The idea behind it is that only 'notes' are kept and transition errors, octave errors and other short events are removed.

Once a selection of the estimations are made or, optionally, other filters are used, the distribution is ready for peak detection. The peak detection algorithm looks for each position where the derivative of

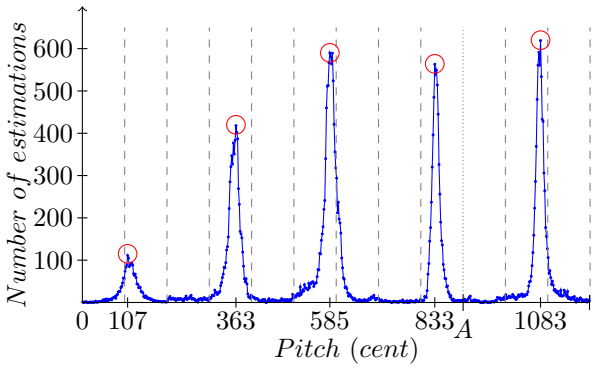
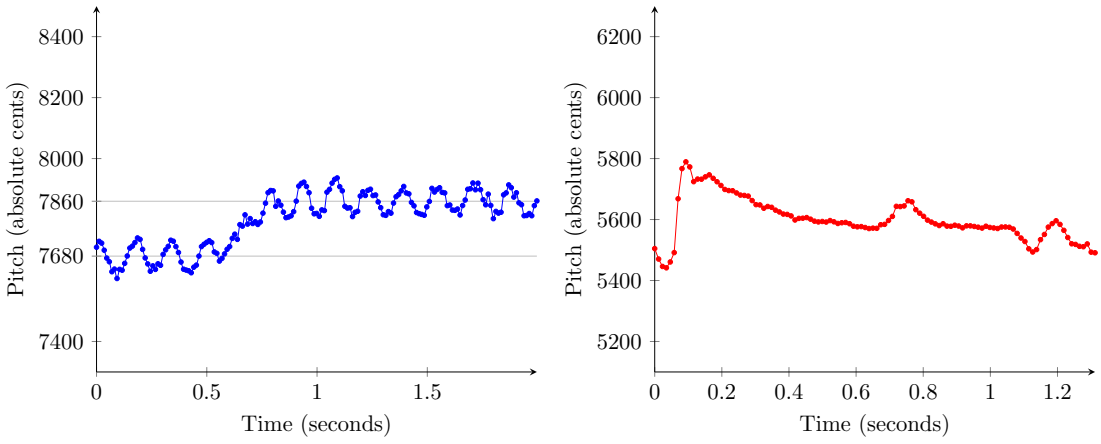


Figure 6: A pitch class histogram with an Indonesian Slendro scale. The circles mark different pitch classes. For reference, the dashed lines represent the Western equal temperament. The pitch class A is marked with a dotted line.

the histogram is zero, and a local height score is calculated with the formula in (1). The local height score s_w is defined for a certain window w, μ_w is the average height in the window, σ_w refers to the standard deviation of the height in the window. The peaks are ordered by their score and iterated, starting from the peak with the highest score. If peaks are found within the window of the current peak, they are removed. Peaks with a local height score lower than a defined threshold are ignored. Since we are looking for pitch classes, the window w wraps around the edges: there is a difference of 20 cent between 1190 cent and 10 cent.

$$s_w = \frac{height - \mu_w}{\sigma_w} \tag{1}$$

Figure 7 shows the local height score function applied to the pitch class histogram shown in Figure 6. The desired leveling effect of the local height score is clear, as the small peak at 107 cents becomes much more defined. The threshold is also shown. In this case, it eliminates the noise at around 250 cents. The noise is caused by the small window size and local height deviations, but it is ignored by setting threshold t. The performance of the peak detection depends on two parameters, namely, the window size and the threshold. Automatic analysis either uses



(a) Western vibrato melograph.

(b) Indian raga pitch gesture.

(c) Western vibrato pitch histogram.

(d) Indian raga pitch gesture histogram.

Figure 5: Visualization of pitch contours of Western and Indian singing; notice the fundamentally different concept of tone. In the western example two distinct pitches are used, they are made clear with the dotted gray lines. In Figure 5c two dotted gray curves are added, they represent the two perceived pitches.

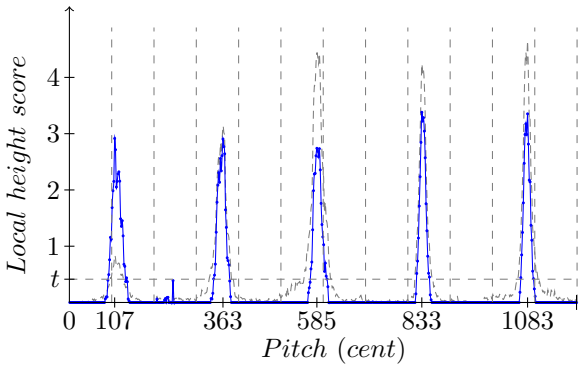


Figure 7: A local height score function used to detect peaks in a pich class histogram. Comparing the original histogram of figure 6 with the local height score shows the leveling effect of the local height score function. The dashed vertical lines represents the Western equal temperament, the dashed horizontal line the threshold t.

a general preset for the parameters or tries to find the most stable setting with an exhaustive search. Optionally gaussian smoothing can be applied to the pitch class histogram, which makes peak detection more straightforward. Manual intervention is sometimes needed, by fiddling with the two parameters a user can quickly browse through several peak detection result candidates.

Once the pitch classes are identified, a pitch class interval matrix can be constructed. This is the fourth type of representation, which is shown in Table 1. The pitch class interval matrix represents the found pitch classes, and shows the intervals between the pitch classes. In our example, a perfect fourth[4], a frequency ratio of 4/3 or 498 cent, is present between pitch class 585 and 1083. This means that a perfect fifth, a frequency ratio of $\frac{2/1}{4/3} = 3/2$ or $1200 - 498 = 702$ cent, is also present[5].

[4]The perfect fourth and other musical intervals are here used in their physical meaning. The physical perfect fourth is sometimes called just fourth, or perfect fourth in just intonation.

[5]See Appendix B to see how ratios translate to cent values.

P.C.	107	364	585	833	1083
107	0	256	478	726	976
364	944	0	221	470	719
585	722	979	0	248	**498**
833	474	730	952	0	250
1083	224	481	**702**	950	0

Table 1: Pitch classes (P.C.) and pitch class intervals, both in cents. The same pentatonic Indonesian slendro is used as in figure 6. A prefect fifth and its dual, a perfect fourth, are marked by a bold font.

2.3 The interface

Most of the capabilities of Tarsos are used through the graphical user interface (Figure 8). The interface provides a way to explore pitch organization within a musical piece. However, the main flow of the process, as described above, is not always as straightforward as the example might suggest. More particularly, in many cases of music from oral traditions, the peaks in the pitch class histogram are not always well-defined (see Section 4). Therefore, the automated peak detection may need manual inspection and further manual fine-tuning in order to correctly identify a songs' pitch organization. The user interface was designed specifically for having a flexible environment where all windows with representations communicate their data. Tarsos has the attractive feature that all actions, like the filtering actions mentioned in Section 2.2, are updated for each window in real-time.

One way to closely inspect pitch distributions is to select only a part of the estimations. In the block diagram of Figure 2, this is represented by the funnel. Selection in time is possible using the waveform view (Figure 8-5). For example, the aim could be a comparison of pitch distributions at the beginning and the end of a piece, to reveal whether a choir lowered or raised its pitch during a performance (see Section 4 for a more elaborate example).

Selection in pitch range is possible and can be combined with a selection in time using the melograph (Figure 8-3). One may select the melodic range such as to exclude pitched percussion, and this could yield a completely different pitch class histogram. This fea-

This is an Author's Original Manuscript of an article whose final and definitive form, the Version of Record, has been published in the Journal of New Music Research (JNMR), August 2013, © Taylor & Francis.
Available online at http://www.tandfonline.com/10.1080/09298215.2013.797999 .

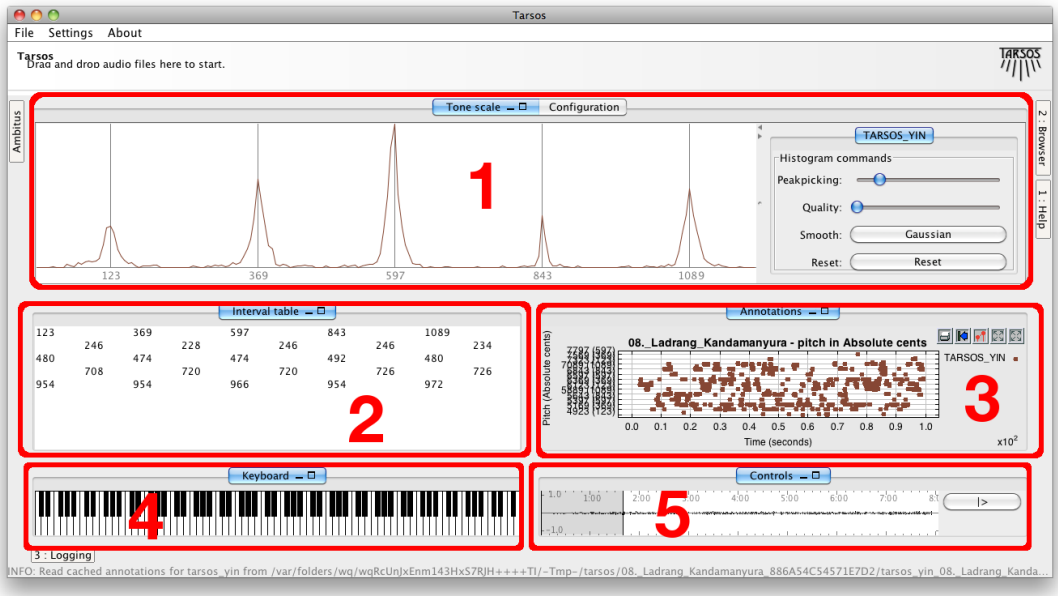


Figure 8: A screenshot of Tarsos with 1 a pitch class histogram, 2 a pitch class interval table, 3 a melograph with pitch estimations, 4 a MIDI keyboard and 5 a waveform.

ture is practical, for example when a flute melody is accompanied with a low-pitched drum and when you are only interested in flute tuning. With the melograph it is also possible to zoom in on one or two notes, which is interesting for studying pitch contours. As mentioned earlier, not all music is organized by fixed pitch classes. An example of such pitch organization is given in Figure 5b, a fragment of Indian music where the estimations contain information that cannot be reduced to fixed pitch classes.

To allow efficient selection of estimations in the time and frequency, they are stored in a kd-tree (Bentley, 1975). Once such a selection of estimations is made, a new pitch histogram is constructed and the pitch class histogram view (Figure 8-1) changes instantly.

Once a pitch class histogram is obtained, peak detection is a logical next step. With the user interface, manual adjustment of the automatically identified peaks is possible. New peak locations can be added and existing ones can be moved or deleted. In order to verify the pitch classes manually, it is possible to click anywhere on the pitch class histogram. This sends a MIDI-message with a pitch bend to synthesize a sound with a pitch that corresponds to the clicked location. Changes made to the peak locations propagate instantly throughout the interface.

The pitch class interval matrix (Figure 8-2) shows all new pitch class intervals. Reference pitches are added to the melograph and MIDI tuning messages are sent (see Section 2.5). The pitch class interval matrix is also interactive. When an interval is clicked on, the two pitch classes that create the interval sound at the same time. The dynamics of the process and the combination of both visual and auditory clues makes manually adjusted, precise peak extraction, and therefore tone scale detection, possible. Finally, the graphical display of a piano keyboard in Tarsos allows us to play in the (new) scale. This feature can be executed on a computer keyboard as well, where notes are projected on keys. Any of the standard MIDI instruments sounds can be chosen.

This is an Author's Original Manuscript of an article whose final and definitive form, the Version of Record, has been published in the Journal of New Music Research (JNMR), August 2013, © Taylor & Francis.
Available online at http://www.tandfonline.com/10.1080/09298215.2013.797999 .

It is possible to shift the pitch class histogram up- or downwards. The data is then viewed as a repetitive, octave based, circular representation. In order to compare scales, it is possible to upload a previously detected scale (see Section 2.5) and shift it, to find a particular fit. This can be done by hand, exploring all possibilities of overlaying intervals, or the best fit can be suggested by Tarsos.

2.4 Real-time capabilities

Tarsos is capable of real-time pitch analysis. Sound from a microphone can be analyzed and immediate feedback can be given on the played or sung pitch. This feature offers some interesting new use-cases in education, composition, and ethnomusicology.

For educational purposes, Tarsos can be used to practice singing quarter tones. Not only the real time audio is analyzed, but also an uploaded scale or previously analyzed file can be listened to by clicking on the interval table or by using the keyboard. Singers or string players could use this feature to improve their intonation regardless of the scale they try to reach.

For compositional purposes, Tarsos can be used to experiment with microtonality. The peak detection and manual adjustment of pitch histograms allows the construction of any possible scale, with the possibility of setting immediate harmonic and melodic auditory feedback. Use of the interval table and the keyboard, make experiments in interval tension and scale characteristics possible. Musicians can tune (ethnic) instruments according to specific scales using the direct feedback of the real-time analysis. Because of the MIDI messages, it is also possible to play the keyboard in the same scale as the instruments at hand.

In ethnomusicology, Tarsos can be a practical tool for direct pitch analysis of various instruments. Given the fact that pitch analysis results show up immediately, microphone positions during field recordings can be adjusted on the spot to optimize measurements.

2.5 Output capabilities

Tarsos contains export capabilities for each step, from the raw pitch estimations until the pitch class interval matrix. The built-in functions can export the data as comma separated text files, charts, TEX-files, and there is a way to synthesize estimations. Since Tarsos is scriptable there is also a possibility to add other export functions or modify the existing functions. The API and scripting capabilities are documented on the Tarsos website: `http://tarsos.0110.be/tag/JNMR`.

For pitch class data, there is a special standardized text file defined by the Scala[6] program. The Scala file format has the `.scl` extension. The Scala program comes with a dataset of over 3900 scales ranging from historical harpsichord temperaments over ethnic scales to scales used in contemporary music. Recently this dataset has been used to find the universal properties of scales (Honingh & Bod, 2011). Since Tarsos can export scala files it is possible to see if the star-convex structures discussed in Honingh & Bod (2011) can be found in scales extracted from real audio. Tarsos can also parse Scala files, so that comparison of theoretical scales with tuning practice is possible. This feature is visualized by the upwards Scala arrow in Figure 2. When a scale is overlaid on a pitch class histogram, Tarsos finds the best fit between the histogram and the scala file.

A completely different output modality is MIDI. The MIDI Tuning Standard defines MIDI messages to specify the tuning of MIDI synthesizers. Tarsos can construct `Bulk Tuning Dump`-messages with pitch class data to tune a synthesizer enabling the user to play along with a song in tune. Tarsos contains the Gervill synthesizer, one of the very few (software) synthesizers that offer support for the MIDI Tuning Standard. Another approach to enable users to play in tune with an extracted scale is to send pitch bend messages to the synthesizer when a key is pressed. Pitch bend is a MIDI-message that tells how much higher or lower a pitch needs to sound in comparison with a standardized pitch. Virtually all synthesizers support pitch bend, but pitch bends operate on MIDI-channel level. This makes it impossible to play

[6] See `http://www.huygens-fokker.org/scala/`

This is an Author's Original Manuscript of an article whose final and definitive form, the Version of Record, has been published in the Journal of New Music Research (JNMR), August 2013, © Taylor & Francis.
Available online at `http://www.tandfonline.com/10.1080/09298215.2013.797999` .

polyphonic music in an arbitrary tone scale.

2.6 Scripting capabilities

Processing many audio files with the graphical user interface quickly becomes tedious. Scripts written for the Tarsos API can automate tasks and offer a possibility to utilize Tarsos' building blocks in entirely new ways. Tarsos is written in Java, and is extendable using scripts in any language that targets the JVM (Java Virtual Machine) like JRuby, Scala[7] and Groovy. For its concurrency support, concise syntax and seamless interoperability with Java, the Scala programming languages are used in example scripts, although the concepts apply to scripts in any language. The number of applications for the API is only limited by the creativity of the developer using it. Tasks that can be implemented using the Tarsos API are for example:

Tone scale recognition: given a large number of songs and a number of tone scales in which each song can be brought, guess the tone scale used for each song. In section 3.4 this task is explained in detail and effectively implemented.

Modulation detection: this task tries to find the moments in a piece of music where the pitch class histogram changes from one stable state to another. For western music this could indicate a change of mode, a modulation. This task is similar as the one described in Lesley Mearns (2011). With the Tarsos API you can compare windowed pitch histograms and detect modulation boundries.

Evolutions in tone scale use: this task tries to find evolutions in tone scale use in a large number of songs from a certain region over a long period of time. Are some pitch intervals becoming more popular than others? In Moelants et al. (2009) this is done for a set of African songs.

Acoustic Fingerprinting: it is theorized in Tzanetakis et al. (2002) that pitch class histograms can serve as an acoustic fingerprint for a song. With the building blocks of Tarsos: pitch detection, pitch class histogram creation and comparison this was put to the test by Six & Cornelis (2012).

The article by Tzanetakis et al. (2002) gives a good overview of what can be done using pitch histograms and, by extension, the Tarsos API. To conclude: the Tarsos API enables developers to quickly test ideas, execute experiments on large sets of music and leverage the features of Tarsos in new and creative ways.

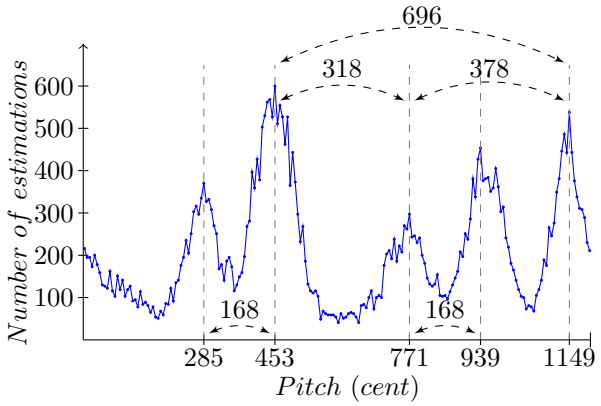
3 Exploring Tarsos' Capabilities Through Case Studies

In what follows, we explore Tarsos' capabilities using case studies in non-Western music. The goal is to focus on problematic issues such as the use of different pitch extractors, music with pitch drift, and last but not least, the analysis of large databases.

3.1 Analysing a Pitch Histogram

We will first consider the analysis of a song that was recorded in 1954 by missionary Scohy-Stroobants in Burundi. The song is performed by a singing soloist, Léonard Ndengabaganizi. The recording was analysed with the YIN pitch detection method and a pitch class histogram was calculated: it can be seen in Figure 9. After peak detection on this histogram, the following pitch intervals were detected: 168, 318, 168, 210, and 336 cents. The detected peaks and all intervals are shown in an interval matrix (see Figure 9). It can be observed that this is a pentatonic division that comprises small and large intervals, which is different from an equal tempered or meantone division. Interestingly, the two largest peaks define a fifth interval, which is made of a pure minor third (318 cents) and a pure major third (378 cents) that lies between the intervals $168 + 210 = 378$ cents). In addition, a mirrored set of intervals is present, based on 168-318-168 cents. This phenomena is also illustrated by Figure 9.

[7] Please do not confuse the general purpose Scala programming language with the tool to experiment with tunings, the Scala program.



P.C.	285	453	771	939	1149
285	0	*168*	486	654	864
453	1032	0	*318*	486	**696**
771	714	882	0	*168*	*378*
939	546	714	1032	0	210
1149	336	504	822	990	0

Figure 9: This song uses an unequally divided pentatonic tone scale with mirrored intervals 168-318-168, indicated on the pitch class histogram. Also indicated is a near perfect fifth consisting of a pure minor and pure major third.

3.2 Different Pitch Extractors

However, Tarsos has the capability to use different pitch extractors. Here we show the difference between seven pitch extractors on a histogram level. A detailed evaluation of each algorithm cannot be covered in this article but can be found in the cited papers. The different pitch extractors are:

- YIN (Cheveigné & Hideki, 2002) (YIN) and the McLeod Pitch Method (MPM), which is described in (McLeod & Wyvill, 2005), are two time-domain pitch extractors. Tarsos contains a platform independent implementation of the algorithms.
- Spectral Comb (SC), Schmitt trigger(Schmitt) and Fast Harmonic Comb (FHC) are described in Brossier (2006). They are available for Tarsos through VAMP-plugins (Cannam, 2008);
- MAMI 1 and MAMI 6 are two versions of the same pitch tracker. MAMI 1 only uses the most present pitch at a certain time, MAMI 6 takes the six most salient pitches at a certain time into account. The pitch tracker is described in Clarisse et al. (2002).

Figure 10 shows the pitch histogram of the same song as in the previous section, which is sung by an unaccompanied young man. The pitch histogram shows a small tessitura and wide pitch classes. However, the general contour of the histogram is more or less the same for each pitch extraction method, five pitch classes can be distinguished in about one-and-a-half octaves, ranging from 5083 to 6768 cent. Two methods stand out. Firstly, MAMI 6 detects pitch in the lower and higher regions. This is due to the fact that MAMI 6 always gives six pitch estimations in each measurement sample. In this monophonic song this results in octave - halving and doubling - errors and overtones. Secondly, the Schmitt method also stands out because it detects pitch in regions where other methods detect a lot less pitches, e.g. between 5935 and 6283 cent.

Figure 11 shows the pitch class histogram for the same song as in Figure 10, now collapsed into one octave. It clearly shows that it is hard to determine the exact location of each pitch class. However, all histogram contours look similar except for the Schmitt method, which results in much less well defined peaks. The following evaluation shows that this is not only the case.

In order to be able to gain some insight into the differences between the pitch class histograms resulting from different pitch detection methods, the following procedure was used: for each song in a data set of more than 2800 songs - a random selection of the mu-

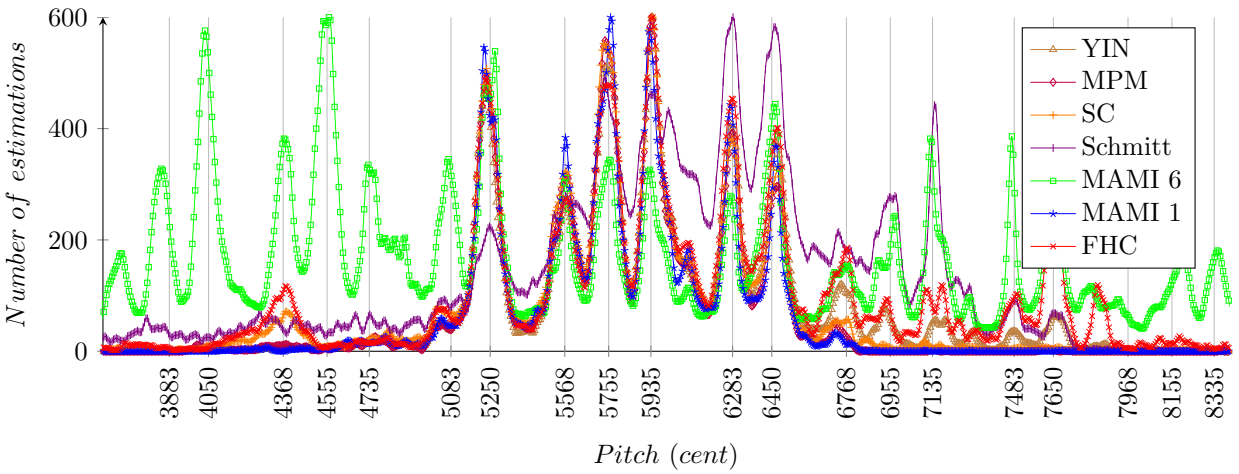


Figure 10: Seven different pitch histograms of a traditional Rwandese song. Five pitch classes repeat every octave. The Schmitt trigger (Schmitt) results in much less well defined peaks in the pitch histogram. MAMI 6 detects much more information to be found in the lower and higher regions, this is due to the fact that it always gives six pitch estimations, even if they are octave errors or overtones.

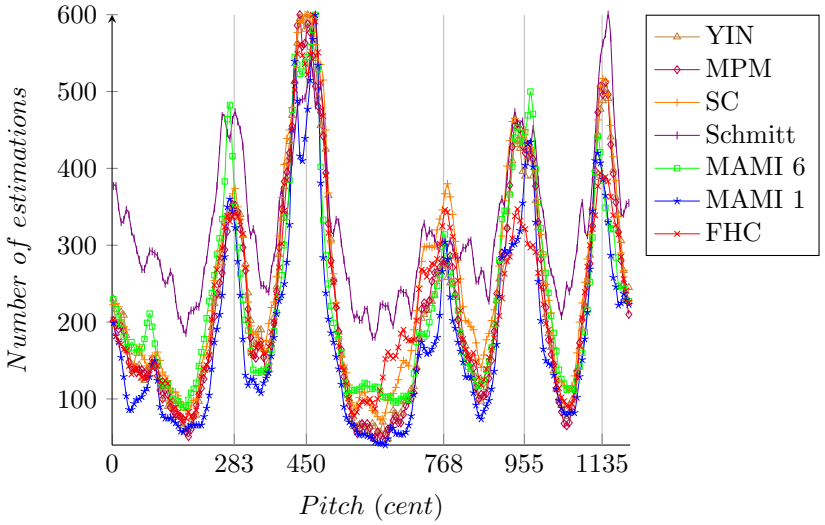


Figure 11: Seven different pitch class histograms of a traditional Rwandese song. Five pitch classes can be distinguished but is clear that it is hard to determine the exact location of each pitch class. The Schmitt trigger (Schmitt) results in a lot less well defined peaks in the pitch class histogram.

	YIN	MPM	Schmitt	FHC	SC	MAMI 1	MAMI 6
YIN	1.00	**0.81**	*0.41*	0.65	0.62	0.69	0.61
MPM	**0.81**	1.00	0.43	0.67	0.64	0.71	0.63
Schmitt	*0.41*	0.43	1.00	0.47	0.53	0.42	0.56
FHC	0.65	0.67	0.47	1.00	0.79	0.67	0.66
SC	0.62	0.64	0.53	0.79	1.00	0.65	0.70
MAMI 1	0.69	0.71	0.42	0.67	0.65	1.00	0.68
MAMI 6	0.61	0.63	0.56	0.66	0.70	0.68	1.00
Average	0.69	0.70	*0.55*	0.70	0.70	0.69	0.69

Table 2: Similarity matrix showing the overlap of pitch class histograms for seven pitch detection methods. The similarities are the mean of 2484 audio files. The last row shows the average of the overlap for a pitch detection method.

sic collection of the Belgian Royal Museum of Central Africa (RMCA) - seven pitch class histograms were created by the pitch detection methods. The overlap - a number between zero and one - between each pitch class histogram pair was calculated. A sum of the overlap between each pair was made and finally divided by the number of songs. The resulting data can be found in Table 2. Here histogram overlap or intersection is used as a distance measure because Gedik & Bozkurt (2010) shows that this measure works best for pitch class histogram retrieval tasks. The overlap $c(h_1, h_2)$ between two histograms h_1 and h_2 with K classes is calculated with equation 2. For an overview of alternative correlation measures between probability density functions see Cha (2007).

$$c(h_1, h_2) = \frac{\sum_{k=0}^{K-1} min(h_1(k), h_2(k))}{max(\sum_{k=0}^{K-1} h_1(k), \sum_{k=0}^{K-1} h_2(k))} \quad (2)$$

The table 2 shows that there is, on average, a large overlap of 81%, between the pitch class histograms created by YIN and those by MPM. This can be explained by the fact that the two pitch extraction algorithms are very much alike: both operate in the time-domain with autocorrelation. The table also shows that Schmitt generates rather unique pitch class histograms. On average there is only 55% overlap with the other pitch class histogram. This performance was already expected during the analysis of one song (above).

The choice for a particular pitch detection method depends on the music and the analysis goals. The music can be monophonic, homophonic or polyphonic, different instrumentation and recording quality all have influence on pitch estimators. Users of Tarsos are encouraged to try out which pitch detection method suits their needs best. Tarsos' scripting API - see section 3.4 - can be helpful when optimizing combinations of pitch detection methods and parameters for an experiment.

3.3 Shifted Pitch Distributions

Several difficulties in analysis and interpretation may arise due to pitch shift effects during musical performances. This is often the case with a capella choirs. Figure 13 shows a nice example of an intentionally raised pitch, during solo singing in the Scandinavian Sami culture. The short and repeated melodic motive remains the same during the entire song, but the pitch raises gradually ending up 900 cents higher than the beginning. Retrieving a scale for the entire song is in this case irrelevant, although the scale is significant for the melodic motive. Figure 14 shows an example where scale organization depends on the characteristics of the instrument. This type of African fiddle, the iningidi, does not use a soundboard to shorten the strings. Instead the string is shortened by the fingers that are in a (floating) position above the string: an open string and three fingers give an tetratonic scale. Figure 12 shows an iningidi being played. This use

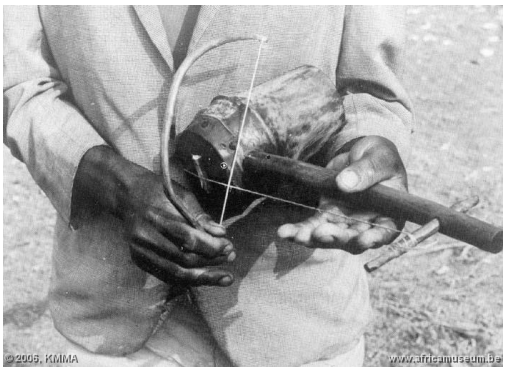


Figure 12: The iningidi, a type of African fiddle. To play the instrument, the neck is held in the palm of the left hand so that the string can be stopped using the second phalanx of the index, middle and ring fingers. Consequently, a total of four notes can be produced.

case shows that pitch distributions for entire songs can be misleading, in both cases it is much more informative to compare the distribution from the first part of the song with the last part. Then it becomes clear how much pitch shifted and in what direction.

Interesting to remark is that these intervals have more or less the same distance, a natural consequence of the distance of the fingers, and that, consequently, not the entire octave tessitura is used. In fact only 600 cents, half an octave, is used. A scale that occurs typically in fiddle recordings, that rather can be seen as a tetrachord. The open string (lowest note) is much more stable than the three other pitches that deviate more, as is shown by the broader peaks in the pitch class histogram. The hand position without soundboard is directly related to the variance of these three pitch classes. When comparing the second minute of the song with the seventh, one sees a clear shift in pitch, which can be explained by the fact the musician changed the hand position a little bit. In addition, another phenomena can be observed, namely, that while performing, the open string gradually loses tension, causing a small pitch lowering which can be noticed when comparing the two fragments. This is not uncommon for ethnic music instruments.

3.4 Tarsos' Scripting applied to Makam Recognition

In order to make the use of scripting more concrete, an example is shown here. It concerns the analysis of Turkish classical music. In an article by Gedik & Bozkurt (2010), pitch histograms were used for - amongst other tasks - makam[8] recognition. The task was to identify which of the nine makams is used in a specific song. With the Tarsos API, a simplified, generalized implementation of this task was scripted in the Scala programming language. The task is defined as follows:

> For a small set of tone scales T and a large set of musical performances S, each brought in one of the scales, identify the tone scale t of each musical performance s automatically.

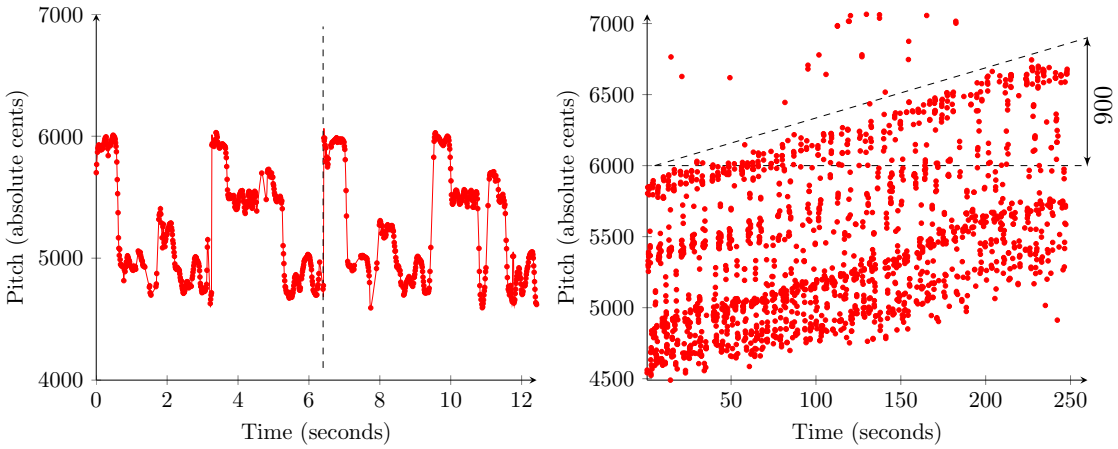
Algorithm 1 Tone scale recognition algorithm

```
1:  T ← constructTemplates()
2:  S ← fetchSongList()
3:  for all s ∈ S do                          ▷ For all songs
4:      O ← {}                                ▷ Initialize empty hash
5:      h ← constructPitchClassHisto(s)
6:      for all t ∈ T do                      ▷ For all templates
7:          o ← calculateOverlap(t, h)
8:          O[s] ← o                          ▷ Store overlap in hash
9:      end for
10:     i ← getFirstOrderedByOverlap(O))
11:     write s "is brought in tone scale" i
12: end for
```

An example of makam recognition can be seen in Figure 15. A theoretical template - the dotted, red line - is compared to a pitch class histogram - the solid, blue line - by calculating the maximum overlap between the two. Each template is compared with the pitch class histogram, the template with maximum overlap is the guessed makam. Pseudocode for this procedure can be found in Algorithm 1.

[8] A makam defines rules for a composition or performance of classical Turkish music. It specifies melodic shapes and pitch intervals.

This is an Author's Original Manuscript of an article whose final and definitive form, the Version of Record, has been published in the Journal of New Music Research (JNMR), August 2013, © Taylor & Francis.
Available online at http://www.tandfonline.com/10.1080/09298215.2013.797999 .



(a) Motive (b) Contour

Figure 13: An a capella song performed by Nils Hotti from the Sami culture shows the gradual intentional pitch change during a song. The melodic motive however is constantly repeated (here shown twice).

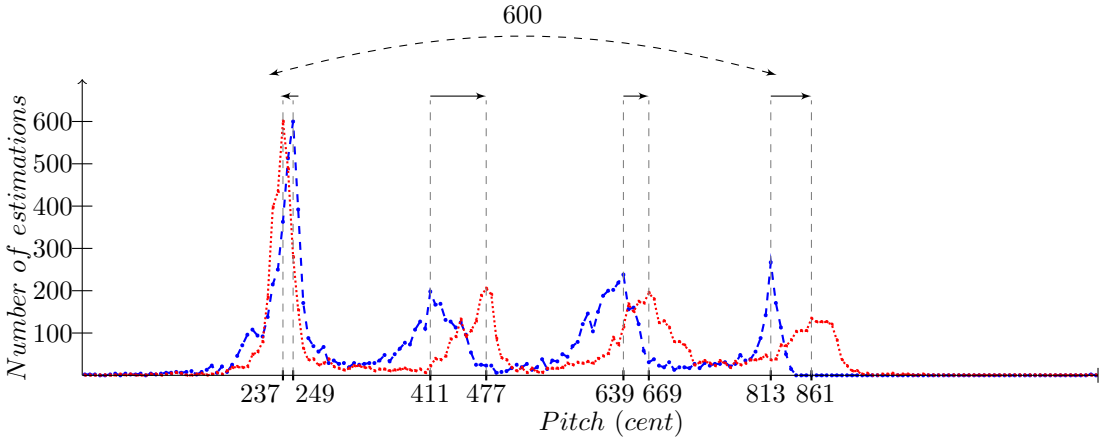


Figure 14: Histogram of an African fiddle song. The second minute of the song is represented by the dashed line, the seventh minute is represented by the dotted line. The lowest, most stable pitch class is the result of the open string. It lost some tension during the piece and started to sound lower. This is in sharp contrast with the other pitch classes that sound higher, due to a change in hand position.

Listing 1: Template construction

```
val makams = List(  "hicaz","huseyni","huzzam","kurdili_hicazar",
           "nihavend","rast","saba","segah","ussak")

var theoreticKDEs = Map[java.lang.String,KernelDensityEstimate]()
makams.foreach{ makam =>
  val scalaFile =  makam + ".scl"
  val scalaObject = new ScalaFile(scalaFile);
  val kde = HistogramFactory.createPichClassKDE(scalaObject,35)
  kde.normalize
  theoreticKDEs = theoreticKDEs + (makam -> kde)
}
```

Makam	**Pitch classes (in cents)**																		
Hicaz		113					384			498	701	792					996		
Huseyni			181		294					498	701				883		996		
Huzzam		113				316			430		701		812						1109
Kurdili Hicazar	90				294					498	701	792					996		
Nihavend				203	294					498	701	792					996		
Rast				203			384			498	701					905		1086	
Saba			181		294			407			701	792					996		
Segah		113				316				498	701			815					1109
Ussak			181		294					498	701	792					996		

Table 3: The nine makams used in the recognition task.

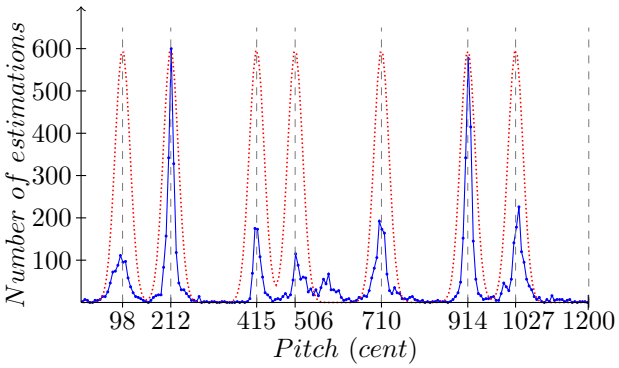


Figure 15: The solid, blue line is a pitch class histogram of a Turkish song brought in the makam Hicaz. The dotted, red line represents a theoretical template of that same Hicaz makam. Maximizing the overlap between a theoretical and an actual pitch class histogram suggests which makam is used.

To construct the tone-scale templates theoretical descriptions of those tone scales are needed, for makams these can be found in Gedik & Bozkurt (2010). The pitch classes are converted to cent units and listed in Table 3. An implementation of *constructTemplates*() in Algorithm 1 can be done as in Listing 1. The capability of Tarsos to create theoretical tone scale templates using Gaussian kernels is used, line 8. Line 7 shows how a scala file containing a tone-scale description is used to create an object with the same information, the height is normalized.

The *calculateOverlap*(t, h) method from line 7 in Algorithm 1 is pitch invariant: it shifts the template to achieve maximum overlap with respect to the pitch class histogram. Listing 2 contains an implementation of the matching step. First a list of audio files is created by recursively iterating a directory and matching each file to a regular expression. Next, starting from line 4, each audio file is processed. The internal implementation of the YIN pitch detection algorithm is used on the audio file and a pitch class histogram is created (line 6,7). On line 10, normalization of the histogram is activated, to make the correlation calculation meaningful. In line 11 to line 15 the created histogram from the audio file is compared with the templates calculated beforehand (in Listing 1). The results are stored, ordered and eventually printed on line 19.

The script ran on Bozkurts data set with Turkish music: with this straightforward algorithm it is possible to correctly identify 39% of makams in a data set of about 800 songs. The results for individual makam recognition vary between 76% and 12% depending on how distinguishable the makam is. If the first three guesses are evaluated, the correct makam is present in 75% of the cases. Obviously, there is room for improvement by using more domain knowledge. A large improvement can be made by taking into account the duration of each pitch class in each template. Bozkurt does this by constructing templates by using the audio itself. A detailed failure analysis falls outside the scope of this article. It suffices to say that practical tasks can be scripted successfully using the Tarsos API.

4 Musicological aspects of Tarsos

Tarsos is a tool for the analysis of pitch distributions. For that aim, Tarsos incorporates several pitch extraction modules, has pitch distribution filters, audio feedback tools, and scripting tools for batch processing of large databases of musical audio. However, pitch distributions can be considered from different perspectives, such as ethnographical studies of scales (Schneider, 2001), theoretical studies in scale analysis (Sethares, 2005), harmonic and tonal analysis (Krumhansl & Shepard, 1979; Krumhansl, 1990), and other structural analysis approaches to music (such as set theoretical and Schenkerian). Clearly, Tarsos does not offer a solution to all these different approaches to pitch distributions. In fact, seen from the viewpoint of Western music analysis, Tarsos is a rather limited tool as it doesn't offer harmonic analysis, nor tonal analysis, nor even statistical analysis of pitch distributions. All of this should be applied together with Tarsos, when needed. Instead, what Tarsos provides is an intermediate level between pitch extraction (done by pitch extractor tools) and music theory.

Listing 2: Makam Recognition

```
val directory = "/home/user/turkish_makams/"
val audio_pattern = ".*.(mp3|wav|ogg|flac)"
val audioFiles = FileUtils.glob(directory,audio_pattern,true).toList

audioFiles.foreach{ file =>
  val audioFile = new AudioFile(file)
  val detectorYin = PitchDetectionMode.TARSOS_YIN.getPitchDetector(audioFile)
  val annotations = detectorYin.executePitchDetection()
  val actualKDE = HistogramFactory.createPichClassKDE(annotations,15);
  actualKDE.normalize
  var resultList = List[Tuple2[java.lang.String,Double]]()
  for ((name, theoreticKDE) <- theoreticKDEs){
      val shift = actualKDE.shiftForOptimalCorrelation(theoreticKDE)
      val currentCorrelation = actualKDE.correlation(theoreticKDE,shift)
      resultList =  (name -> currentCorrelation) :: resultList
  }
  //order by correlation
  resultList = resultList.sortBy{_._2}.reverse
  Console.println(file + " is brought in tone scale " + resultList(0)._1)
}
```

Makam	Number of songs	Correct guesses	Percentage of correct guesses
Kurdili Hicazar	91	32	35.16%
Huseyni	64	8	12.50%
Nihavend	75	27	36.00%
Segah	111	43	38.74%
Saba	81	50	61.73%
Huzzam	62	15	24.19%
Rast	118	23	19.49%
Ussak	102	54	52.94%
Hicaz	68	52	76.47%
Total	772	304	39.69%

Table 4: Results of the makam recognition task, using theoretical intervals, on the Bozkurt data set with Turkish music.

The major contribution of Tarsos is that it offers an easy to use tool for pitch distribution analysis that applies to all kinds of music, including Western and non-Western. The major contribution of Tarsos, so to speak, is that it offers pitch distribution analysis without imposing a music theory. In what follows, we explain why such tools are needed and why they are useful.

4.1 Tarsos and Western music theoretical concepts

Up to recently, musical pitch is often considered from the viewpoint of a traditional music theory, which assumes that pitch is stable (e.g. vibrato is an ornament of a stable pitch), that pitch can be segmented into tones, that pitches are based on octave equivalence, and that octaves are divided into 12 equal-sized intervals of each 100 cents, and so on. These assumptions have the advantage that music can be reduced to symbolic representations, a written notation, or notes, whose structures can be studied at an abstract level. As such, music theory has conceptualized pitch distributions as chords, keys, modes, sets, using a symbolic notation.

So far so good, but tools based on these concepts may not work for many nuances of Western music, and especially not for non-Western music. In Western music, tuning systems have a long history. Proof of this can be found in tunings of historical organs, and in tuning systems that have been explored by composers in the 20th century (cf. Alois Haba, Harry Partch, Ivo Darreg, and Lamonte Young). Especially in non-Western classical music, pitch distributions are used that radically differ from the Western theoretical concepts, both in terms of tuning, as well as in pitch occurrence, and in timbre. For example, the use of small intervals in Arab music contributes to nuances in melodic expression. To better understand how small pitch intervals contribute to the organization of this music, we need tools that do no assume octave divisions in 12 equal-sized intervals (see Gedik & Bozkurt (2010)). Other types of music do not have octave equivalence (cf. the Indonesian gamalan), and also some music work with modulated pitch. For example, Henbing & Leman (2007) describe classical Chinese guqin music which uses tones that contain sliding patterns (pitch modulations), which form a substantial component of the tone and consider it as a succession of prototypical gestures. Krishnaswamy (2004b,a) introduces a set of 2D melodic units, melodic atoms, in describing Carnatic (South-Indian classical) music. They represent or synthesize the melodic phrase and are not bound by a scale type. Hence, tools based on Western common music theoretical conceptions of pitch organization may not work for this type of music.

Oral musical traditions (also called ethnic music) provide a special case since there is no written music theory underlying the pitch organization. An oral culture depends on societal coherence, interpersonal influence and individual musicality, and this has implications on how pitch gets organized. Although oral traditions often rely on a peculiar pitch organization, often using a unique system of micro-tuned intervals, it is also the case that instruments may lack a fixed tuning, or that tunings may strongly differ from one instrument to the other, or one region to the other. Apparently, the myriad of ways in which people succeed in making sense out of different types of pitch organization can be considered as cultural heritage that necessitates a proper way of documentation and study (Moelants et al., 2009; Cornelis et al., 2010).

Several studies attempt at developing a proper approach to pitch distributions. Gómez & Bonada (2008) look for pitch gestures in European folk music as an additional aspect to pitch detection. Moving from tone to scale research, Chordia & Rae (2007) acknowledges interval differences in Indian classical music, but reduces to a chromatic scale for similarity analysis and classification techniques. Sundberg & Tjernlund (1969) developed, already in 1969, an automated method for extracting pitch information from monophonic audio for assembling the scale of the spilåpipa by frequency histograms. Bozkurt (2008); Gedik & Bozkurt (2010) build a system to classify and recognize Turkish maqams from audio files using overall frequency histograms to characterize the maqams scales and to detect the tonic centre. Maqams contain intervals of different sizes, often not compatible with the chromatic scale, but partly relying on smaller intervals. Moelants et al. (2007) fo-

cuses on pitch distributions of especially African music that deals with a large diversity of irregular tuning systems. They avoid a priori pitch categories by using a quasi-continuous rather than a discrete interval representation. In Moelants et al. (2009) they show that African songs have shifted more and more towards Western well temperament from 1950s to 1980s.

To sum up, the study of pitch organization needs tools that go beyond elementary concepts of the Western music theoretical canon (such as octave equivalence, stability of tones, equal temporal scale, and so on). This is evident from the nuances of pitch organization in Western music, in non-Western classical music, as well as in oral music cultures. Several attempts have been undertaken, but we believe that a proper way of achieving this is by means of a tool that combines audio-based pitch extraction with a generalized approach to pitch distribution analysis. Such a tool should be able to automatically extract pitch from musical audio in a culture-independent manner, and it should offer an approach to the study of pitch distributions and its relationship with tunings and scales. The envisioned tool should be able to perform this kind of analysis in an automated way, but it should be flexible enough to allow a musicologically grounded manual fine-tuning using filters that define the scope at which we look at distributions. The latter is indeed needed in view of the large variability of pitch organization in music all over the world. Tarsos is an attempt at supplying such a tool. One the one hand, Tarsos tries to avoid music theoretical concepts that could contaminate music that doesn't subscribe the constraints of the Western music theoretical canon. On the other hand, the use of Tarsos is likely to be too limited, as pitch distributions may further draw upon melodic units that may require an approach to segmentation (similar to the way segmented pitch relates to notes in Western music) and further gestural analysis (see the references to the studies mentioned above).

4.2 Tarsos pitfalls

The case studies from section 3 illustrate some of the capabilities of Tarsos as tool for the analysis of pitch distributions. As shown Tarsos offers a graphical interface that allows a flexible way to analyse pitch, similar to other editors that focus on sound analysis (Sonic Visualizer, Audacity, Praat). Tarsos offers support for different pitch extractors, real-time analysis (see section 2.4), and has numerous output capabilities (See section 2.5). The scripting facility allows us to use of Tarsos' building blocks in unique ways efficiently.

However, Tarsos-based pitch analysis should be handled with care. The following three recommendations may be taken into account: First of all, one cannot extract scales without *considering the music itself*. Pitch classes that are not frequently used, won't show up clearly in a histogram and hence might be missed. Also not all music uses distinct pitch classes: the Chinese and Indian music traditions have been mentioned in this case. Because of the physical characteristics of the human voice, voices can glide between tones of a scale, which makes an accurate measurement of pitch not straightforward. It is recommended to zoom in on the estimations in the melograph representation for a correct understanding.

Secondly, analysis of *polyphonic recordings should be handled with care* since current pitch detection algorithms are primarily geared towards monophonic signals. Analysis of homophonic singing for example may give incomplete results. It is advisable to try out different pitch extractors on the same signal to see if the results are trustworthy.

Finally, Schneider (2001) recognizes the use of "pitch categories" but warns that, especially for complex inharmonic sounds, *a scale is more than a one dimensional series of pitches* and that spectral components need to be taken into account to get better insights in tuning and scales. Indeed, in recent years, it became clear that the timbre of tones and the musical scales in which these tones are used, are somehow related (Sethares, 2005). The spectral content of pitch (i.e. the timbre) determines the perception of consonant and dissonant pitch intervals, and therefore also the pitch scale, as the latter is a reflection of the preferred melodic and harmonic combinations of pitch. Based on the principle of minimal dissonance in pitch intervals, it is possible to derive pitch scales from spectral properties of the sounds and principles of auditory interference (or critical bands). Schwartz

& Purves (2004) argue that perception is based on the disambiguation of action-relevant cues, and they manage to show that the harmonic musical scale can be derived from the way speech sounds relate to the resonant properties of the vocal tract. Therefore, the annotated scale as a result of the precise use of Tarsos, does not imply the assignment of any characteristic of the music itself. It is up to the user to correctly interprete of a possible scale, a tonal center, or a melodic development.

4.3 Tarsos - Future work

The present version of Tarsos is a first step towards a tool for pitch distribution analysis. A number of extensions are possible.

For example, given the tight connection between timbre and scale, it would be nice to select a representative tone from the music and transpose it to the entire scale, using a phase vocoder. This sound sample and its transpositions could then be used as a sound font for the MIDI synthesizer. This would give the scale a more natural feel compared to the general MIDI device instruments that are currently present.

Another possible feature is tonic detection. Some types of music have a well-defined tonic, e.g. in Turkish classical music. It would make sense to use this tonic as a reference pitch class. Pitch histograms and pitch class histograms would then not use the reference frequency defined in appendix B but a better suited, automatically detected reference: the tonic. It would make the intervals and scale more intelligible.

Tools for comparing two or more scales may be added. For example, by creating pitch class histograms for a sliding window and comparing those with each other, it should be possible to automatically detect modulations. Using this technique, it should also be possible to detect pitch drift in choral, or other music.

Another research area is to extract features on a large data set and use the pitch class histogram or interval data as a basis for pattern recognition and cluster analysis. With a time-stamped and geo-tagged musical archive, it could be possible to detect geographical or chronological clusters of similar tone scale use.

On the longer term, we plan to add representations of other musical parameters to Tarsos as well, such as rhythmic and instrumental information, temporal and timbral features. Our ultimate goal is to develop an objective albeit partial view on music by combining those three parameters within an easy to use interface.

5 Conclusion

In this paper, we have presented Tarsos, a modular software platform to extract and analyze pitch distributions in music. The concept and main features of Tarsos have been explained and some concrete examples have been given of its usage. Tarsos is a tool in full development. Its main power is related to its interactive features which, in the hands of a skilled music researcher, can become a tool for exploring pitch distributions in Western as well as non-Western music.

6 Bibliography

Bentley, J. L. (1975, september). Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, *18*(9), 509–517.

Bozkurt, B. (2008, March). An Automatic Pitch Analysis Method for Turkish Maqam Music. *Journal of New Music Research (JNMR)*, *37*(1), 1–13.

Brossier, P. (2006). *Automatic Annotation of Musical Audio for Interactive Applications*. Academisch proefschrift, Queen Mary University of London, UK.

Cannam, C. (2008). *The Vamp Audio Analysis Plugin API: A Programmer's Guide*. `http://vamp-plugins.org/guide.pdf`.

Cannam, C., Landone, C., & Sandler, M. (2010). Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Open Source Software Competition, ACM*.

Cha, S.-h. (2007). Comprehensive Survey on Distance / Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, *1*(4), 300–307.

Cheveigné, A. de & Hideki, K. (2002). YIN, a Fundamental Frequency Estimator for Speech and Music. *The Journal of the Acoustical Society of America*, *111*(4), 1917–1930.

Chordia, P. & Rae, A. (2007). Raag Recognition Using Pitch-Class and Pitch-Class Dyad Distributions. In *Proceedings of the 8th International Symposium on Music Information Retrieval (ISMIR 2007).*

Clarisse, L. P., Martens, J. P., Lesaffre, M., Baets, B. D., Meyer, H. D., & Leman, M. (2002). An Auditory Model Based Transcriber of Singing Sequences. In *Proceedings of the 3th International Symposium on Music Information Retrieval (ISMIR 2002)* (pp. 116–123).

Cornelis, O., Lesaffre, M., Moelants, D., & Leman, M. (2010, April). Access to ethnic music: Advances and perspectives in content-based music information retrieval. *Signal Processing*, *90*(4), 1008–1031.

Gedik, A. C. & Bozkurt, B. (2010). Pitch-frequency Histogram-based Music Information Retrieval for Turkish Music. *Signal Processing*, *90*(4), 1049–1063.

Gómez, E. & Bonada, J. (2008). Automatic Melodic Transcription of Flamenco Singing. In *Proceedings of 4th Conference on Interdisciplinary Musicology (CIM 2008).*

Helmholtz, H. von & Ellis, A. J. (1912). *On the Sensations of Tone as a Physiological Basis for the Theory of Music* (translated and expanded by Alexander J. Ellis, 2nd English dr.) [Book]. Longmans, Green, London.

Henbing, L. & Leman, M. (2007). A Gesture-based Typology of Sliding-tones in Guqin Music. *Journal of New Music Research (JNMR)*, *36*(2), 61–82.

Honingh, A. & Bod, R. (2011). In Search of Universal Properties of Musical Scales. *Journal of New Music Research (JNMR)*, *40*(1), 81–89.

Klapuri, A. (2003, nov.). Multiple Fundamental Frequency Estimation Based on Harmonicity and Spectral Smoothness. *IEEE Transactions on Speech and Audio Processing*, *11*(6), 804 - 816.

Krishnaswamy, A. (2004a). Melodic Atoms for Transcribing Carnatic Music. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR 2004)* (p. 345-348).

Krishnaswamy, A. (2004b). Multi-dimensional Musical Atoms in South-Indian Classical Music. In *Proceedings of the 8th International Conference on Music Perception & Cognition (ICMPC 2004).*

Krumhansl, C. L. (1990). Tonal Hierarchies and Rare Intervals in Music Cognition. *Music Perception*, *7*(3), 309-324.

Krumhansl, C. L. & Shepard, R. N. (1979). Quantification of the Hierarchy of Tonal Functions Within a Diatonic Context. *Journal of Experimental Psychology: Human Perception and Performance*(5), 579-594.

Lesley Mearns, S. D., Emmanouil Benetos. (2011). Automatically Detecting Key Modulations in J.S. Bach Chorale Recordings. In *Proceedings of the Sound Mucic and Computing Conference (SMC 2011).*

McLeod, P. & Wyvill, G. (2005). A Smarter Way to Find Pitch. In *Proceedings of the International Computer Music Conference (ICMC 2005).*

Moelants, D., Cornelis, O., & Leman, M. (2009). Exploring african tone scales. In *Proceedings of the 10th International Symposium on Music Information Retrieval (ISMIR 2009).*

Moelants, D., Cornelis, O., Leman, M., Matthé, T., Hallez, A., Caluwe, R. D., et al. (2007). Problems and Opportunities of Applying Data- and Audio-Mining Techniques to Ethnic Music. *Journal of Intangible Heritage*, *2*, 57–69.

Nobutaka, O., Miyamoto, K., Kameoka, H., Roux, J. L., Uchiyama, Y., Tsunoo, E., et al. (2010). Harmonic and Percussive Sound Separation and Its Application to MIR-Related Tasks. In Z. W. Ras & A. Wieczorkowska (red.), *Advances in Music Information Retrieval* (Dl. 274). Springer.

Schneider, A. (2001). Sound, Pitch, and Scale: From "Tone Measurements" to Sonological Analysis in Ethnomusicology. *Ethnomusicology*, *45*(3), 489–519.

Schwartz, D. A. & Purves, D. (2004, november). Pitch is Determined by Naturally Occurring Periodic Sounds. *Hearing Research*, *194*(1), 31–46.

Sethares, W. (2005). *Tuning Timbre Spectrum Scale* (2e dr.). Springer.

Six, J. & Cornelis, O. (2012). A Robust Audio Fingerprinter Based on Pitch Class Histograms - Applications for Ethnic Music Archives. In *Proceedings of the Folk Music Analysis conference (FMA 2012).*

Sundberg, J. & Tjernlund, P. (1969). Computer Measurements of the Tone Scale in Performed Music by Means of Frequency Histograms. *STL-QPS*, *10*(2-3), 33–35.

Tzanetakis, G., Ermolinskyi, A., & Cook, P. (2002). Pitch Histograms in Audio and Symbolic Music Information Retrieval. In *Proceedings of the 3th International Symposium on Music Information Retrieval (ISMIR 2002)* (pp. 31–38).

Appendix A Pitch Representation

Since different representations of pitch are used by Tarsos and other pitch extractors this section contains definitions of and remarks on different pitch and pitch interval representations.

For humans the perceptual distance between 220Hz and 440Hz is the same as between 440Hz and 880Hz. A pitch representation that takes this logarithmic relation into account is more practical for some purposes. Luckily there are a few:

MIDI *Note Number*

The MIDI standard defines note numbers from 0 to 127, inclusive. Normally only integers are used but any frequency f in Hz can be represented with a fractional note number n using equation 3.

$$n = 69 + 12\log_2(\frac{f}{440}) \tag{3}$$

$$n = 12 \times \log_2(\frac{f}{r}) \; ; \; r = \frac{440}{2^{(69/12)}} = 8.176\text{Hz} \tag{4}$$

Rewriting equation 3 to 4 shows that MIDI note number 0 corresponds with a reference frequency of 8.176Hz which is C_{-1} on a keyboard with A_4 tuned to 440Hz. It also shows that the MIDI standard divides the octave in 12 equal parts.

To convert a MIDI note number n to a frequency f in Hz one of the following equations can be used.

$$f = 440 \times 2^{(n-69)/12} \tag{5}$$

$$f = r \times 2^{(n/12)} \text{ with } r = 8.176\text{Hz} \tag{6}$$

Using pitch represented as fractional MIDI note numbers makes sense when working with MIDI instruments and MIDI data. Although the MIDI note numbering scheme seems oriented towards western pitch organization (12 semitones) it is conceptually equal to the cent unit which is more widely used in ethnomusicology.

Cent

Helmholtz & Ellis (1912) introduced the nowadays widely accepted cent unit. To convert a frequency f in Hz to a cent value c relative to a reference frequency r also in Hz.

$$c = 1200 \times \log_2(\frac{f}{r}) \tag{7}$$

With the same reference frequency r equations 7 and 4 differ only by a constant factor of exactly 100. In an environment with pitch representations in MIDI note numbers and cent values it is practical to use the standardized reference frequency of 8.176Hz.

To convert a frequency f in Hz to a cent value c relative to a reference frequency r also in Hz.

$$f = r \times 2^{(c/1200)} \tag{8}$$

Savart & Millioctaves

Divide the octave in 301.5 and 1000 parts respectively, which is the only difference with cents.

A.1 Pitch Ratio Representation

Pitch ratios are essentially pitch intervals, an interval of one octave, 1200 cents equal to a frequency ratio of 2/1. To convert a ratio t to a value in cent c:

$$c = \frac{1200 \ln(t)}{\ln(2)} \tag{9}$$

The natural logarithm, the logarithm base e with e being Euler's number, is noted as ln. To convert a value in cent c to a ratio t:

$$t = e^{\frac{c \ln(2)}{1200}} \tag{10}$$

Further discussion on cents as pitch ratios can be be found in appendix B of Sethares (2005). There it is noted that:

> There are two reasons to prefer cents to ratios: Where cents are added, ratios are multiplied; and it is always obvious which of two intervals is larger when both are expressed in cents. For instance, an interval of a just fifth, followed by a just third is (3/2)(5/4) = 15/8, a just seventh. In cents, this is 702 + 386 = 1088. Is this larger or smaller than the Pythagorean seventh 243/128? Knowing that the latter is 1110 cents makes the comparison obvious.

A.2 Conclusion

The cent unit is mostly used for pitch interval representation while the MIDI key and Hz units are used mainly to represent absolute pitch. The main difference between cent and fractional MIDI note numbers is the standardized reference frequency. In our software platform Tarsos we use the exact same standardized reference frequency of 8.176Hz which enables us to use cents to represent absolute pitch and it makes conversion to MIDI note numbers trivial. Tarsos also uses cents to represent pitch intervals and ratios.

Appendix B Audio material

Several audio files were used in this paper to demonstrate how Tarsos works and to clarify musical concepts. In this appendix you can find pointers to these audio files.

The thirty second excerpt of the musical example used throughout chapter 2 can be downloaded on `http://tarsos.0110.be/tag/JNMR` and is courtesy of: WERGO/Schott Music & Media, Mainz, Germany, `www.wergo.de` and Museum Collection Berlin. Ladrang Kandamanyura (slendro pathet manyura) is track eight on Lestari - *The Hood Collection, Early Field Recordings from Java* - `SM 1712 2`. It is recorded in 1957 and 1958 in Java.

The yoiking singer of Figure 13 can be found on a production released on the label Caprice Records in the series of Musica Sveciae Folk Music in Sweden. The album is called Jojk CAP 21544 CD 3, Track

This is an Author's Original Manuscript of an article whose final and definitive form, the Version of Record, has been published in the Journal of New Music Research (JNMR), August 2013, © Taylor & Francis.
Available online at `http://www.tandfonline.com/10.1080/09298215.2013.797999` .

No 38 Nila, hans svager/His brother-in-law Nila.

The API example (section 3.4) was executed on the data set by Bozkurt. This data set was also used in (Gedik & Bozkurt, 2010). The Turkisch song brought in the makam Hicaz from Figure 15 is also one of the songs in the data set.

For the comparison of different pitch trackers on pitch class histogram level (section 3.2) a subset of the music collection of the Royal Museum for Central Africa (RMCA, Tervuren, Belgium) was used. We are grateful to the RMCA for providing access to its unique archive of Central African music. A song from the RMCA collection was also used in section 3.1. It has the tape number MR.1954.1.18-4 and was recorded in 1954 by missionary Scohy-Stroobants in Burundi. The song is performed by a singing soloist, Léonard Ndengabaganizi. Finally the song with tape number MR.1973.9.41-4, also from the collection of the RMCA, was used to show pitch shift within a song (Figure 14). It is called *Kana nakunze* and is recorded by Jos Gansemans in Mwendo, Rwanda in the year 1973.