# From Music to Visuals – and Back

2D Fourier Transform applied on Music Piano Roll Plots

Reinhold Behringer
School of Computing and Creative Technology
Leeds Metropolitan University
Leeds, United Kingdom
r.behringer@leedsmet.ac.uk

***Abstract*—When a 2D Fourier Transform is applied to piano roll plots which are often used in sequencer software, the resulting 2D graphic is a novel music visualization which reveals internal musical structure. This visualization converts the set of musical notes from the notation display in the piano roll plot to a display which shows structure over time and spectrum within a set musical time period. The transformation is reversible, which means that it also can be used as a novel interface for editing music. The concept of this visualization is demonstrated by software which was written for using MIDI files and creating the visualization with the Fast Fourier Transform (FFT) algorithm. This software demonstrates the live real-time display of this visualization in replay of MIDI files or by music input through a connected MIDI keyboard. The resulting display is independent of pitch transformation or tempo. This visualization approach can be used for musicology studies, for music fingerprinting, comparing composition styles, and for a new creative composition method.**

***Keywords-component; MIDI; music; visualisation; FFT; 2D Fourier Transform;***

## I. Introduction

One of the interesting challenges in the creative realm is to explore the relationship between music and visuals. There have been many approaches for "transforming" music into 2-dimensional (2D) visuals (images, videos), and while they started out with manual transformations (e.g. [1] ), there are now many approaches for automatic transformation (e.g. [2]). There are fewer approaches for the reverse transformation (from visuals to music), mostly involving sonification rather than outputting music (e.g. [3]). Also many of the prevalent visualization tools use sound rather than music as input (e.g. media players), and therefore, such visualizations are influenced by instrument timbre and human performance in addition to the music itself. A visualization which would solely take into account the abstract music notation as its input would provide insight into the general character of music structures rather than of a particular music recording.

The Fourier Transformation [4] is a tool which detects periodicity in signals. It is of great use in analyzing audio and sound, as it provides the frequency spectrum of a sound during a short period (typically 100ms). If a set of these spectra is plotted vertically (low frequencies at bottom, high at top) over a horizontal time line, one obtains a 2D spectrogram. Such a spectrogram shows the frequencies (notes) and harmonics over time, equivalent to a piano roll plot used in computer music sequencer software, but it does not show or encode higher-level internal musical structure and longer-term periodicity.

A Fourier Transformation (FT) in general reveals periodicities in signals, but can also be used to transform a general non-periodic signal into its Fourier components. This leads to the idea that an FT approach also could be applied to musical structure itself, in which case a longer time period would have to be covered than with the FT of a typical spectrum computation. The input signal for such a longer time-window (1-dimensional) FT could be the audio signal itself. Such an FT spectrum would show the classical FT spectrum for high frequency components (which are equivalent to music notes themselves), but is also expected to show low-frequency structure (< 10 Hz) of the change of such notes. The computational requirements for this are relatively high, due to the long time window that would need to be covered (>0.5 sec) for the computation of the FT. A much simpler approach would be to solely consider musical events over time as an input for this "slow" FT, instead of computing the full frequency spectra. This leads to the idea of using 2D piano roll plots as input: they are basically 2D spectrograms, showing the distribution of musical notes over time within a set time window. If a 2D FT algorithm is applied onto this plot, then the resulting 2D FT would show features related to both the distribution of notes over time and over pitch. In effect, this is a 2D FT applied on a time-plot of 1D FTs (the spectra).

This approach of computing an FT of (the logarithm of) an FT has been already developed in the 1960s [5], introducing the terms *alanysis* (sic), *cepstrum* (sic), and *quefrency* (sic). This technique has been used in signal processing applications such as speech recognition, but has not been applied to music.

We have begun an explorative research and development to investigate the use of this 2D FT in visualizing music. Hereby we are limiting our efforts to the input of abstract music notation such as MIDI files instead of audio – this avoids the need to also compute the spectrograms. We have begun to develop software for automatic real-time computation of the 2D FT on piano roll plots, obtained from MIDI files and from live play through a keyboard. In this paper we describe our first experiments with the 2D FT, show some early results, and present an outlook on the possible use of this approach.

## II. The Fourier Transform (FT)

### A. *Theoretical Background*

The basic equation for the one-dimensional discrete FT, using complex notation, is given in equation (1). Herby, $x_n$ is the discretely sampled wave input. The equations are written here in the complex form.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi nk/N} \tag{1}$$

For the 2D discrete FT, the summation has to be performed across the two axis of the input array. 2D FT is commonly used in image processing, for image analysis, compression, and filtering. The equation for computing the resulting FT on an array with points $x_{m,n}$ is given in equation (2).

$$X_{k,l} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{m,n} e^{-i2\pi\left(\frac{mk}{M}+\frac{nl}{N}\right)} \tag{2}$$

While the input of the FT can be just an array of **real** numbers with zero imaginary part, the resulting FT is in general an array of **complex** numbers, each of which also can be shown as an array with magnitude and phase. For visualisation, usually the magnitude is the meaningful variable, but in order to keep the proper time line relationship of the input features, the phase is also essential.

A fast algorithm for computing FTs is the Fast Fourier Transform (FFT) algorithm developed in the 1960s [6]. It reduces the $N^2$ complexity for the computation of a 2D FT to $N{\cdot}\log N$ complexity. This algorithm is used in many applications where spectra are computed, and we use it in our software for computing the 2D FT plot of the piano roll plots.

### B. *Interpretation of 2D FT on Piano Plot Rolls*

When applied to a 2D piano roll plot, the input of the 2D FT is the quantized grid (time, pitch) of the roll plot image. Where in a conventional FT applied to the sound wave, the input values are the values of the sampled signal, and in the case of 2D FT the input values are the loudness values of the notes in the quantized 2D grid. The two axes of the resulting 2D FT plot have the following meaning: the horizontal axis represents the rhythmic structure of the music, given in 1/time units as "low frequencies". Small values of the horizontal axis represent "long" notes and/or pauses, indicating low degree of change in the timeline structure of the music, whereas a higher value of the horizontal axis describes a faster changing pace of the notes. The maximum value of the horizontal axes describes the highest possible time resolution of the input plot. The vertical axis of the 2D FT plot refers to the chord structure of the music and the music intervals. A high value on this axis means that note intervals are small and possibly densely clustered. Low values on this axis mean larger intervals. In our approach for the piano roll plot, the input resolution of the frequency axis is one semitone, but this can also be generalized and extended to microtones, simply by setting a finer quantisation of the vertical axis of the piano roll plot.

## III. Experiments with 2D FT on Piano Plot Rolls

Before our own software development process began, we did a few experiments, using existing software for computing the 2D FT on images. For this, an existing software code [7] was chosen, because it provided the necessary capabilities and the code base for our own customization.

In order to explore the feasibility of using the 2D FT approach for music visualization, a few initial experiments were conducted: a music rendition of Gustav Mahler's "Urlicht" (digitized by the author from score [8]) was loaded into the sequencer software Cakewalk® Sonar (v.8.52), and a screenshot of an excerpt of the piano roll plot was produced (see Figure 1), with distracting features (guide lines) being removed from the original input plot. The color of the note bars indicates different instruments, the intensity (darkness) of the color signifies note velocity (a measure of note attack, which can here be taken for note loudness/intensity).

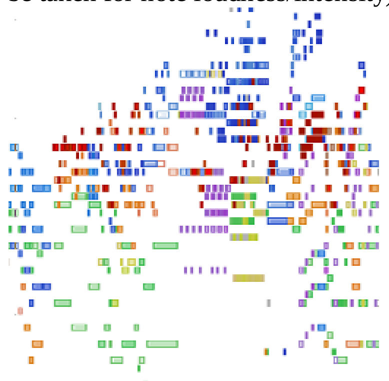
Figure 1.Screenshot of sequencer piano roll plot of Gustav Mahler's "Urlicht".

The 2D FT was then computed on this image by the FFT algorithm, and the magnitude plot was displayed (see Figure 2, left).

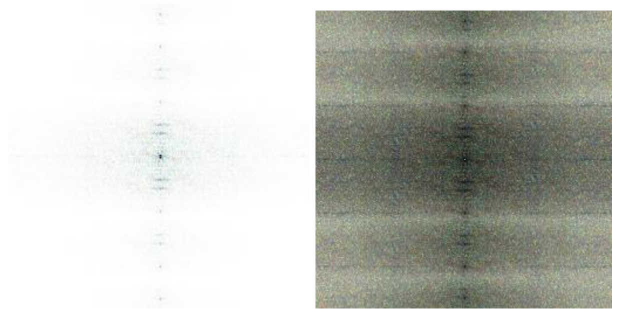
Figure 2. Left: FT magnitude of piano roll plot of Gustav Mahler's "Urlicht". Right: the same plot with logarithmic scaling of intensities.

In order to see more directly the point symmetry around the origin (0;0), the 2D FT plot is shown shifted by ½ of its dimensions, so that the origin appears in the image center. This 2D FT plot, normalized to intensity values between 0 and 255, shows several clearly visible structures around the central vertical axis, which indicate the use of certain intervals. Besides these main features, there are also more features which are more clearly visible in a logarithmic plot of the 2D FT intensities (see Figure 2, right).

In addition to magnitude, the FT also produces a phase plot (see Figure 3). This phase determines the spatial relationship of each point in the image. It appears to show no discernible structure, but it is essential for the inversion of the transformation: together with phase and magnitude of the FT plots, the original image (piano roll plot) can be reproduced identically without any visual loss of visual features.

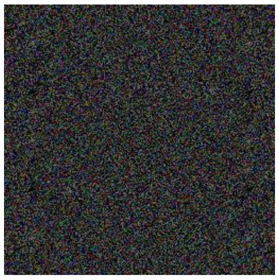

Figure 3. Phase plot of FT of Mahler's "Urlicht".

Inverting the FT re-creates the original image of the piano roll plot. This property of being invertible is used in image processing applications where editing operations can be applied in the 2D FT domain. For the application in the music realm, it is noteworthy that the 2D FT contains all music features from the original input. This makes it possible that the 2D FT plane can be used for editing and modifying its data – which then can lead to new music results when being inverted back.

Problematic is the fact that the phase plot needs to be preserved in order to re-compute the correct original input. In an experiment, we set the imaginary parts of the complex 2D FT plot to zero and only kept the real part of the FT, before then inverting it back. The result was that the inverted plot showed the original piano roll plot, overlaid by a spatially reflected plot version (Figure 4, left). When removing the phase by assigning the full magnitude to the real part and setting the imaginary part to 0, the resulting inversion of the 2D FT shows no resemblance to the original piano roll plot (see Figure 4, right). Therefore, the phase needs to be included properly.

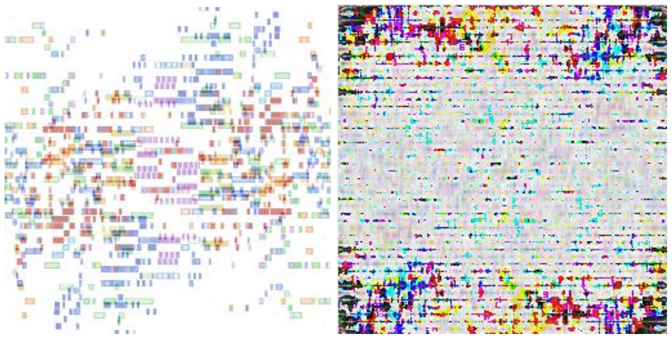

Figure 4. Left: inverse 2D FT with imaginary part set to zero. Right: inverse 2D FT with phase set to zero.

These initial experiments with the application of the 2D FT directly on images of the piano roll plots have been done with color images. In the original sequencer, the different colors of the note bars indicate different instruments, of which there are 33 in individual tracks. The 2D FT algorithm, however, only separates the three basic colors (red, green, blue) in the RGB encoded images. Therefore, the instruments are not treated individually but "mixed" together, and as a result there is no clear color distinction in the 2D FT plots. In principle, the 2D FT would need to be applied to each instrument separately in order to properly reflect all the features contributed by each instrument. This can be achieved if a different input data structure would be used: instead of an RGB image, an array with a set of subfields should be employed, one field for each instrument. Despite these limitations, these initial experiments with the image-based FT were encouraging to further exploration of this 2D FT approach in music.

## IV. The Custom Software for 2D FT

Based on the existing FFT implementation [7], we wrote further software which would allow to directly create a piano roll plot and create the 2D FT in real-time continuously. The piano roll plot would only provide the essential visual elements and would enable the creation of a freely configurable and extendible format for storing the data. The selected development platform is Microsoft's Visual Studio®, the code is written in C# and uses .NET functionality. The MIDI functionality is embedded by using Leslie Sanford's C# MIDI toolkit [9], which provides functionality for reading and playing MIDI sequences and for connecting MIDI devices. The software allows loading a MIDI file, which can then be played through the built-in MS software synthesizer ("Microsoft GS Wavetable Synth".). Furthermore, the software allows manual live play through an external keyboard.

For creating the piano roll plot, an internal 128x128 array of INT values stores the piano roll plot. The vertical axes is directly the pitch value of the MIDI note (with 64 = C). The values in each array element are the MIDI velocity values. At a later stage of the development, these values will be computed from volume (MIDI #7) and expression (MIDI #11) values, which more properly reflect the changing loudness of a MIDI sound. In the current software version, the decision was made to interpret the time axis as "musical time", that is in relation to musical measures and tempo. This allows a tempo-invariant computation of the FT. The user can choose what the overall range of the visible piano roll plot should be: 1 bar, 2 bars, or 4 bars. The tempo determines the speed with which the piano roll is scrolled. It is set from the MIDI file, but also can be scaled. New note events are added into the array at the right end of the plot, and with the scrolling of the roll plot to the left, parts on the left end are removed. This occurs at every new note event and at pre-determined times triggered by a timer, which can be set to every 1/4, 1/8, 1/16, or 1/32 note. This trigger also causes the FT to be computed on the current state of the piano roll plot.

The graphics of the piano roll plot and the magnitude of the 2D FT are shown enlarged as 256x256 graphics. For greater detail, they can be "zoomed" in at the centre by a factor of 2. Furthermore, the FT can be shown in shifted view with the origin at the centre instead of the bottom left.

The user can also play live through either an on-screen keyboard or an external MIDI (USB) keyboard. The tempo for the plot creation can be set (default is 120 bpm), and a click is played at each beat, so that the user can play in sync with the beat.

## V. Systematic Experiments

With our custom software, it is possible to study some of the features of the 2D FT in more detail, by playing live on the keyboard certain music input patterns. The following figures are screenshots of the software and show both the piano roll plot (left) and the 2D FT plot (right). Figure 5 shows the 2D FT of a single note: multiple vertical lines. This is caused by the step at the beginning of the note, equivalent to the multiple Fourier components of a rectangular signal. These multiple components disappear, once the step disappears and the note is shown solid (Figure 6).

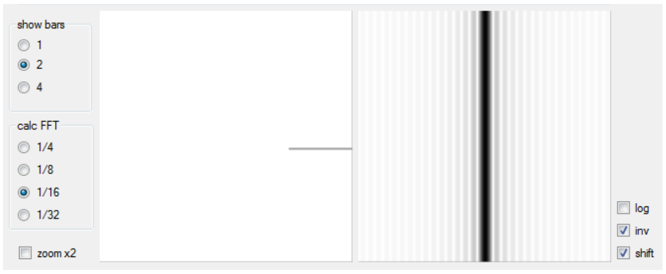
show bars 1 [✓ 2] 4
calc FFT 1/4 1/8 [✓ 1/16] 1/32
zoom x2
log [✓ inv] [✓ shift]

Figure 5. Beginning of single long note. The sudden jump of the volume causes the multiple components in the 2D FT.

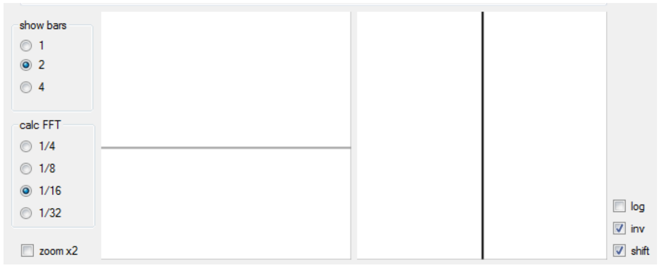
show bars 1 [✓ 2] 4
calc FFT 1/4 1/8 [✓ 1/16] 1/32
zoom x2
log [✓ inv] [✓ shift]

Figure 6. Single long note with indefinite duration causes straight vertical line in center of 2D FT.

If the note is played in a rhythm, the line pattern in the 2D FT indicates the rhythmic properties (Figure 7). This takes also into account emphasis on individual notes.

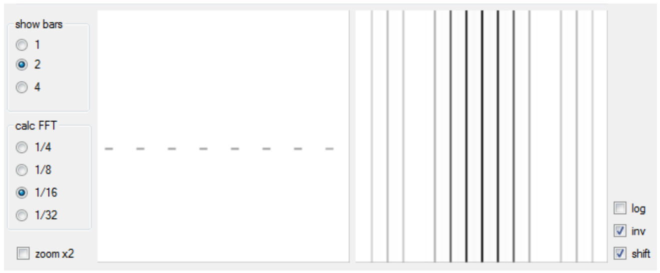
show bars 1 [✓ 2] 4
calc FFT 1/4 1/8 [✓ 1/16] 1/32
zoom x2
log [✓ inv] [✓ shift]

Figure 7. Short note with quarter beat causes set of equidistant lines in 2D FT, indicating a fixed rhythm.

Notes played at intervals cause a structure along the vertical axis of the 2D FT (see Figure 8). Note here also the alternating play of the notes, which results in the shifted dashing of the vertical lines. Interesting is the 2D FT structure in Figure 9, where a simple up-and-down-sweep causes a complex structure.

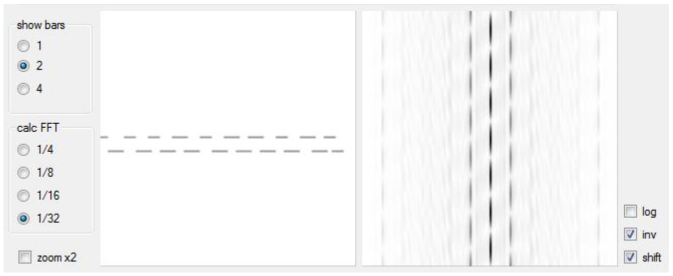
show bars 1 [✓ 2] 4
calc FFT 1/4 1/8 1/16 [✓ 1/32]
zoom x2
log [✓ inv] [✓ shift]

Figure 8. Alternating tones every quarter, a fifth apart. This causes vertical lines and shifted line segments in the 2D FT.

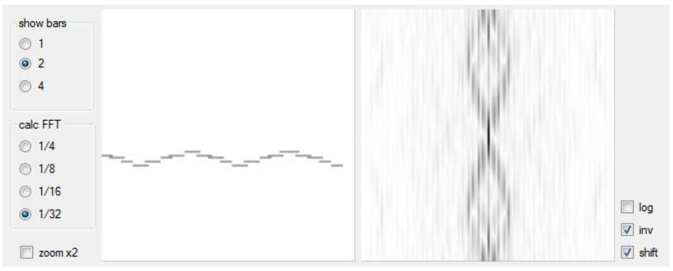
show bars 1 [✓ 2] 4
calc FFT 1/4 1/8 1/16 [✓ 1/32]
zoom x2
log [✓ inv] [✓ shift]

Figure 9. Up-and-down-sweep with regular note duration.

## VI. Conclusions and Outlook

We believe that the 2D FT presented in this paper can be used in music for various purposes: it provides a reversible way of detecting features and structure in music, allowing new ways of visualization and composition of music. Also, the 2D FT provides a unique fingerprint of music, based on the music alone (tempo and pitch invariant), and this is also suitable for further musicological analysis. More work needs to be done in creating intuitive tools for interacting with this 2D FT and allowing to harness and comprehend all its features.

## References

[1] Oskar Fischinger, Oskar Fischinger Archive. www.oskarfischinger.org

[2] Stephen Malinowski, “The music animation machine”, www.musanim.com, 2006.

[3] Peter B.L. Meijer, “The vOICe”, 2012, http://bit.ly/43xnsk

[4] Jean Baptiste Joseph Fourier, J.B.J., Théorie Analytique de la Chaleur, Paris, 1822.

[5] B.P.Bogert, M.J.R.Healy, and John Wilder Tukey: “The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo Autocovariance, Cross-Cepstrum and Saphe Cracking”. Proceedings of the Symposium on Time Series Analysis (M. Rosenblatt, Ed) Chapter 15, 209-243. New York: Wiley, 1963

[6] James W.Cooley and John W. Tukey, ”An algorithm for the machine calculation of complex Fourier series,” Math. Comput. 19, 297-301, 1965.

[7] V.A.Bharadi, “2D FFT of an image in C#”. *The CODE Project.* 2009 http://bit.ly/yHGkVt

[8] Gustav Mahler. “Symphony No.2”. Philharmonia Scores, PH395

[9] Sanford, L. “C# MIDI Toolkit”. *The CODE Project.* 2007. http://bit.ly/wI3tst