



A Sound Visualization Project

From music to animated Kandinsky painting

By Zeinab Esmaeelzadeh
Sommersemester 2018
MidTerm Presentation

The project is divided in 2 parts:

Part I :

Sound analysis/Signal processing

Input :

- Analogue
Sound
Noise
Microphone
(AUX input)
- Digital:
Midi

Output:

Matrices of numbers:

- Columns: Selected feature/Properties:
Pitch,ADSR,Velocity,Selected
frequency,etc
- Rows: Slice of time considering beat or
feasibility of CPU

Part II:

Visualization in R language

Input:

Matrices —▶ R data frame

Output:

Pictures similar to Kandinsky's
Paintings for each feasible slice of
time.

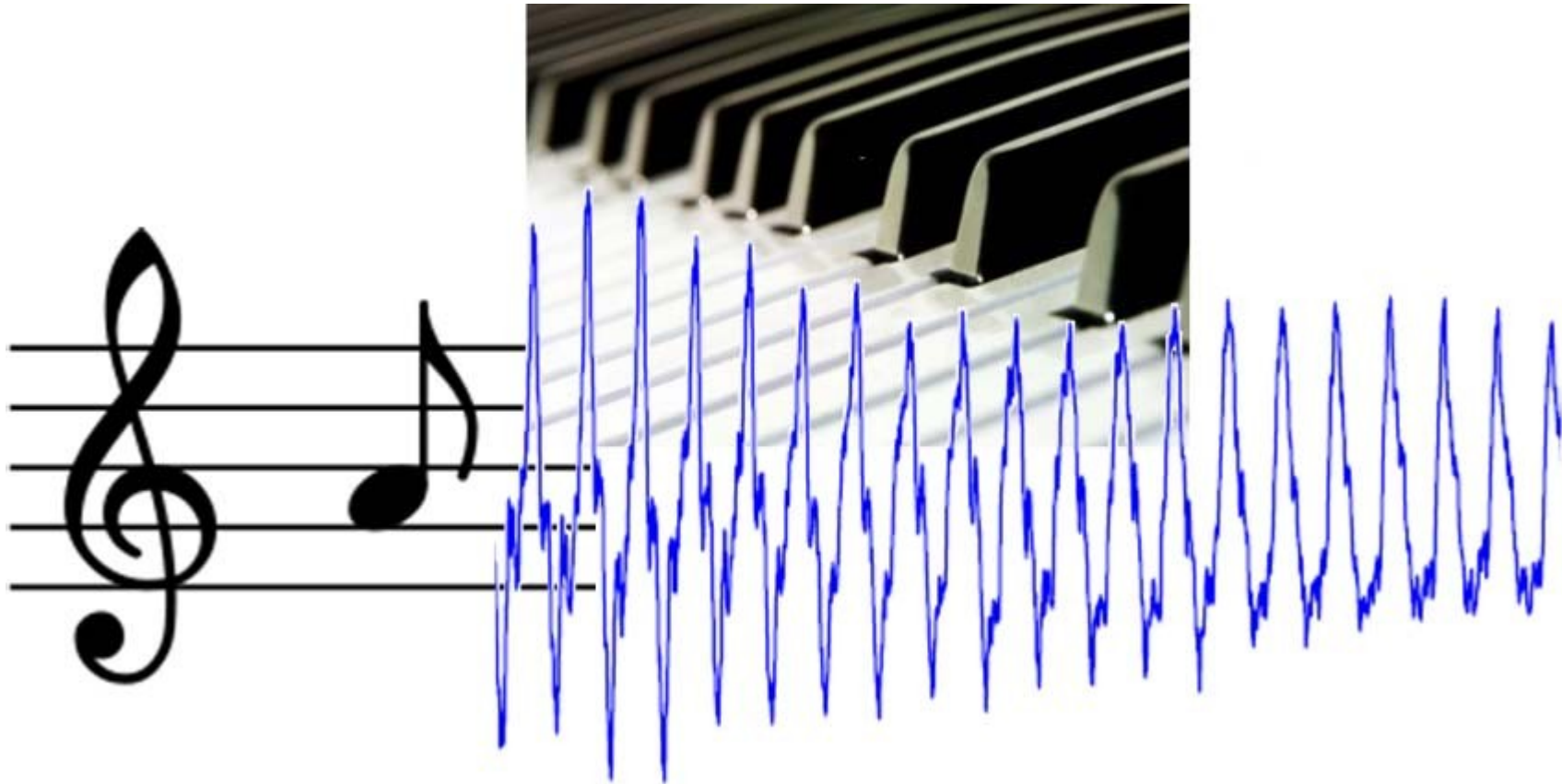
(modified Kandinsky R package)

And then animating pictures in video

Introduction to music representation



- 1.1 Sheet Music Representations
- 1.2 Symbolic Representations
- 1.3 Audio Representation

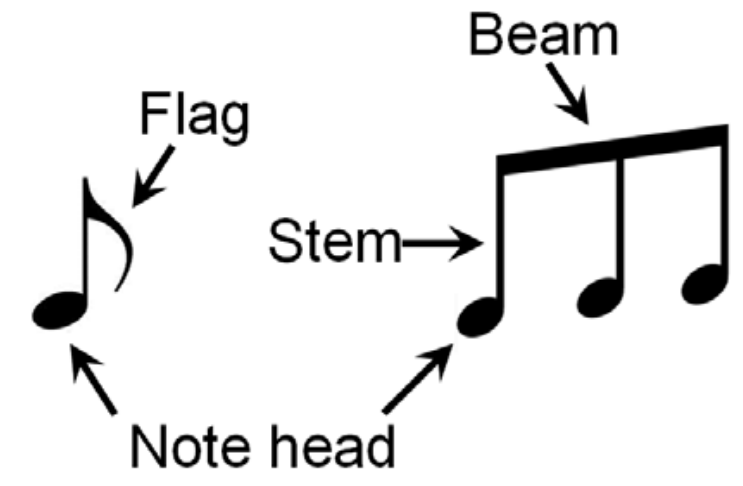


1.1 Sheet Music Representations

Allegro con brio (♩ = 108)

The image shows a musical score for piano in 2/4 time, featuring a forte (ff) dynamic and a 'Ped.' marking with a flower symbol. The score is written on two staves, treble and bass clef, with a grand staff bracket on the left. The key signature has two flats (B-flat and E-flat). The tempo is marked 'Allegro con brio' with a quarter note equal to 108 beats per minute. The music consists of six measures. The first measure has a forte (ff) dynamic marking. The second and fourth measures have a 'Ped.' marking with a flower symbol. The third and fifth measures have a fermata over the final note. The sixth measure has a fermata over the final note. The bass line features a sequence of eighth notes in the first measure, followed by a half note in the second measure, and a sequence of eighth notes in the third measure. The treble line features a sequence of eighth notes in the first measure, followed by a half note in the second measure, and a sequence of eighth notes in the third measure.

1.1 Sheet Music Representations



1.1 Sheet Music Representations

Allegro con brio. $\text{♩} = 108.$

Flauti.

Oboi.

Clarineti in B.

Fagotti.

Corni in Es.

Trombe in C.

Timpani in C.G.

Allegro con brio. $\text{♩} = 108.$

Violino I.

Violino II.

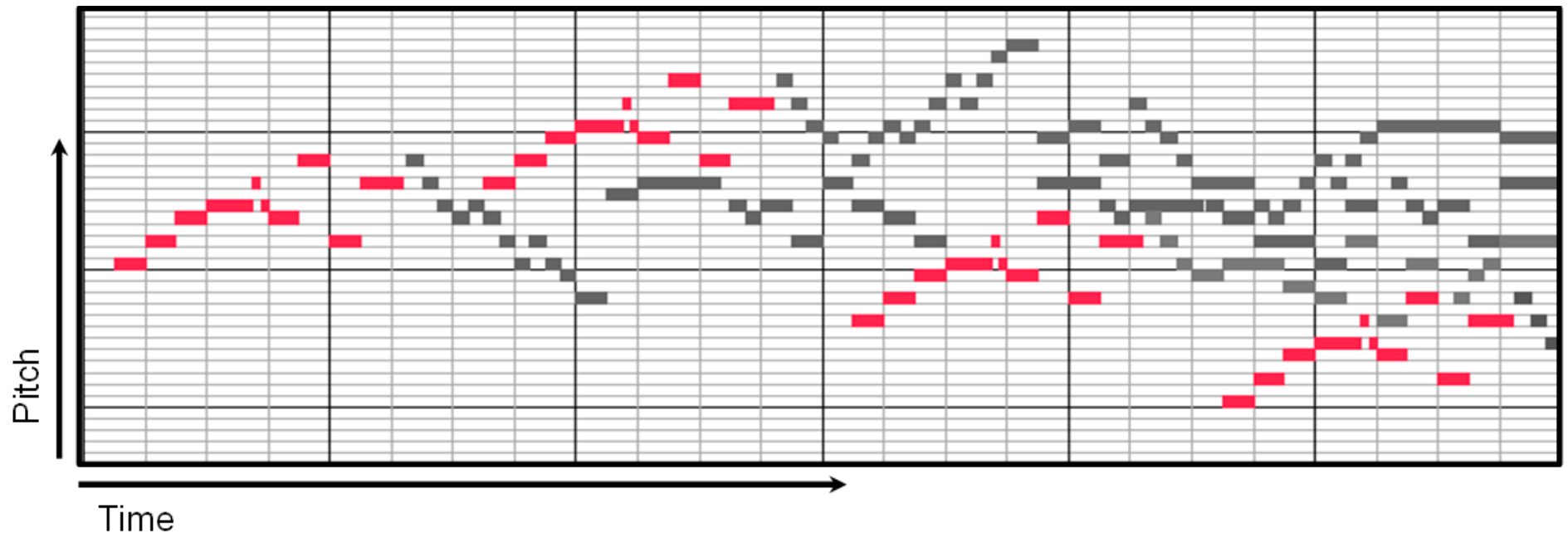
Viola.

Violoncello.

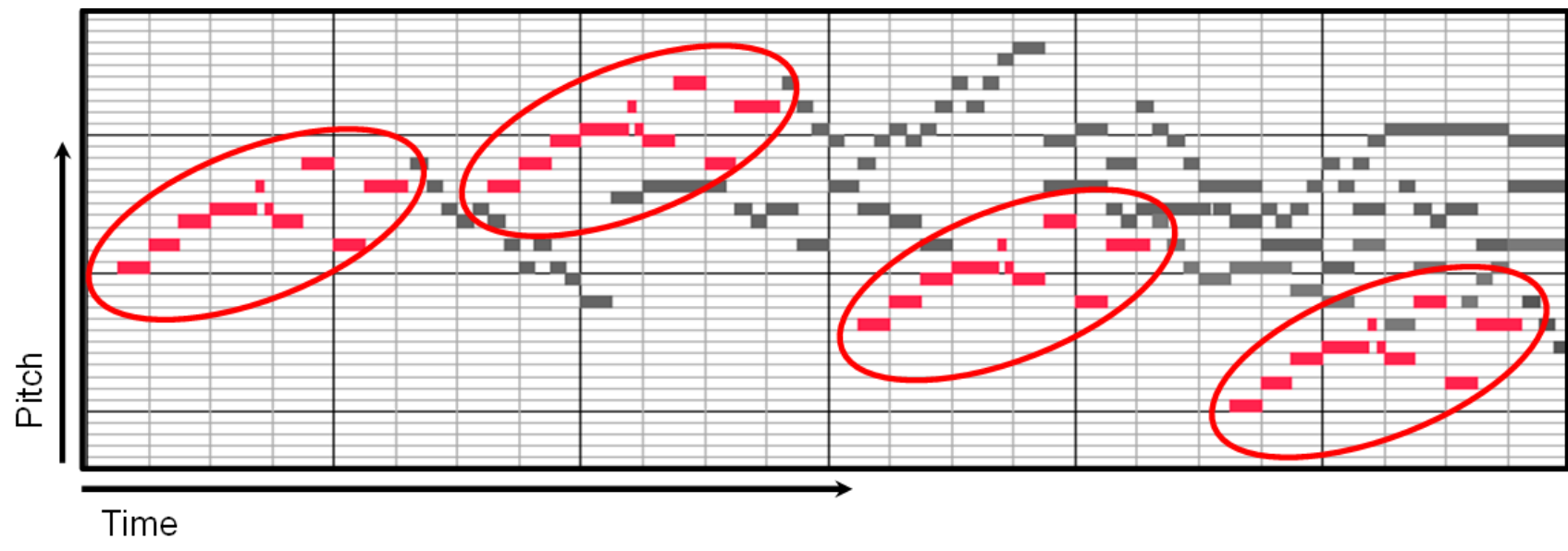
Basso.

The image displays a page of sheet music for a symphony. It features two systems of staves. The first system includes staves for Flauti, Oboi, Clarineti in B, Fagotti, Corni in Es, and Trombe in C. The second system includes staves for Timpani in C.G., Violino I, Violino II, Viola, Violoncello, and Basso. The tempo is marked 'Allegro con brio' with a quarter note equal to 108 beats per minute. The key signature has two flats. The woodwind and brass parts are mostly rests, while the string parts have active lines with dynamic markings like *ff* and *p*.

1.2 Symbolic Representations



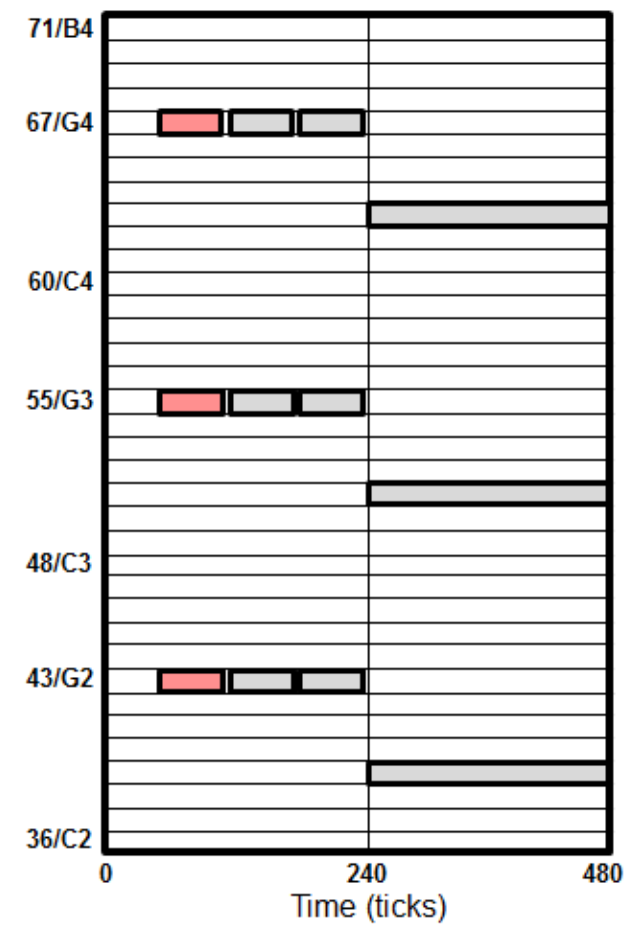
1.2 Symbolic Representations



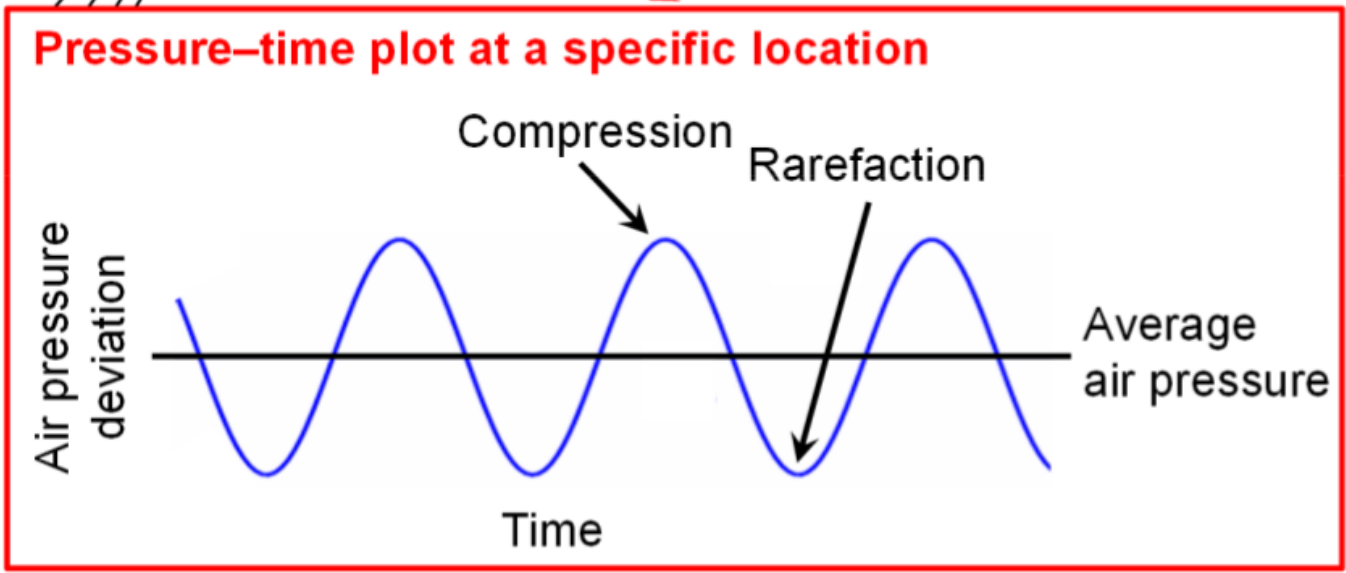
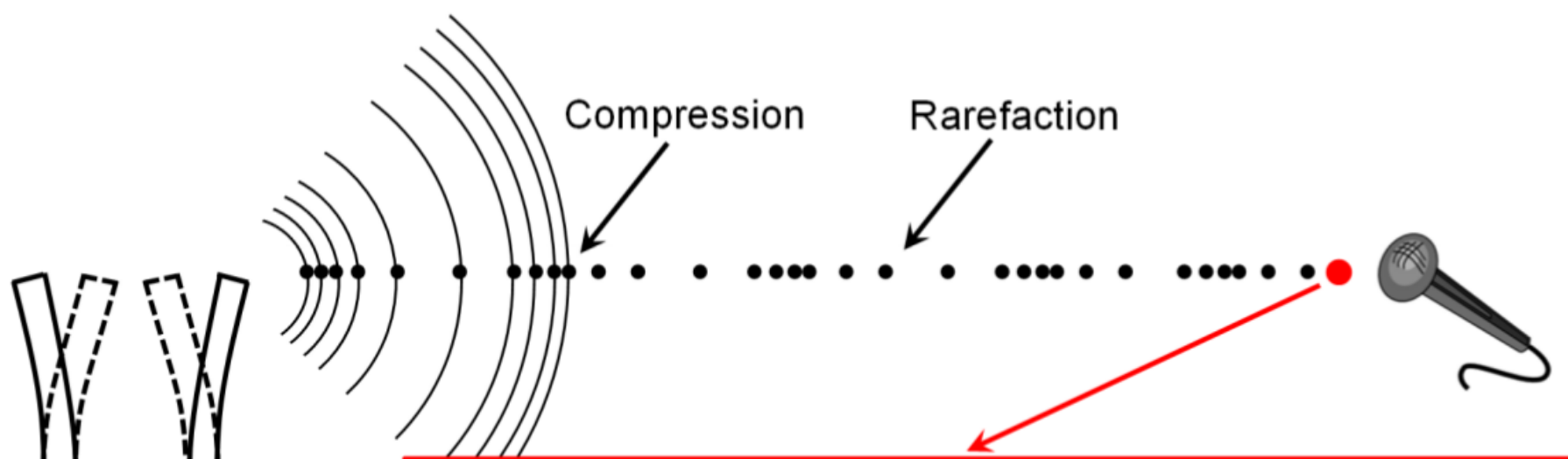
1.2 Symbolic Representations



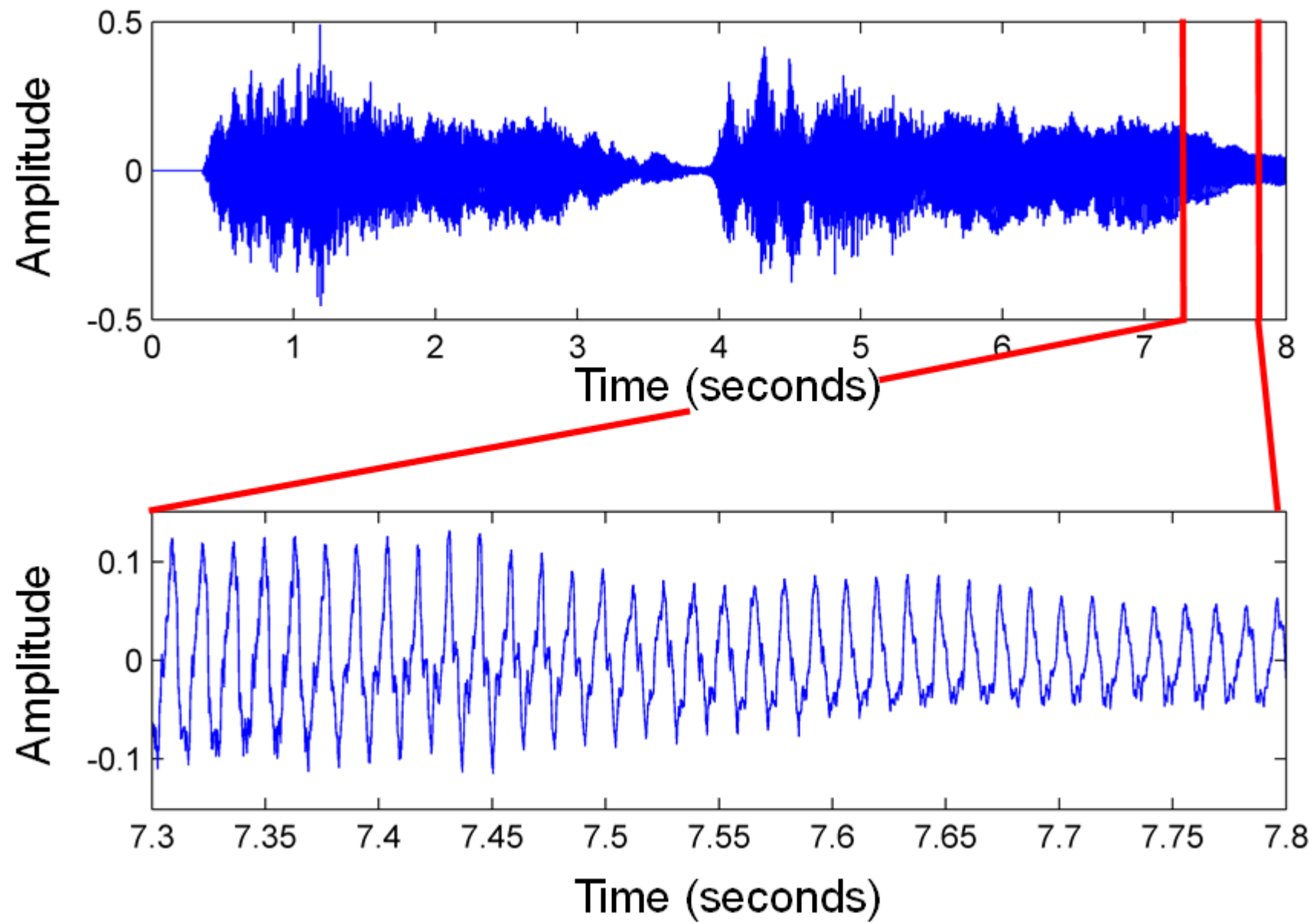
Time (Ticks)	Message	Channel	Note Number	Velocity
60	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	67	100
0	NOTE ON	1	55	100
0	NOTE ON	2	43	100
55	NOTE OFF	1	67	0
0	NOTE OFF	1	55	0
0	NOTE OFF	2	43	0
5	NOTE ON	1	63	100
0	NOTE ON	2	51	100
0	NOTE ON	2	39	100
240	NOTE OFF	1	63	0
0	NOTE OFF	2	51	0
0	NOTE OFF	2	39	0



1.3 Audio Representation

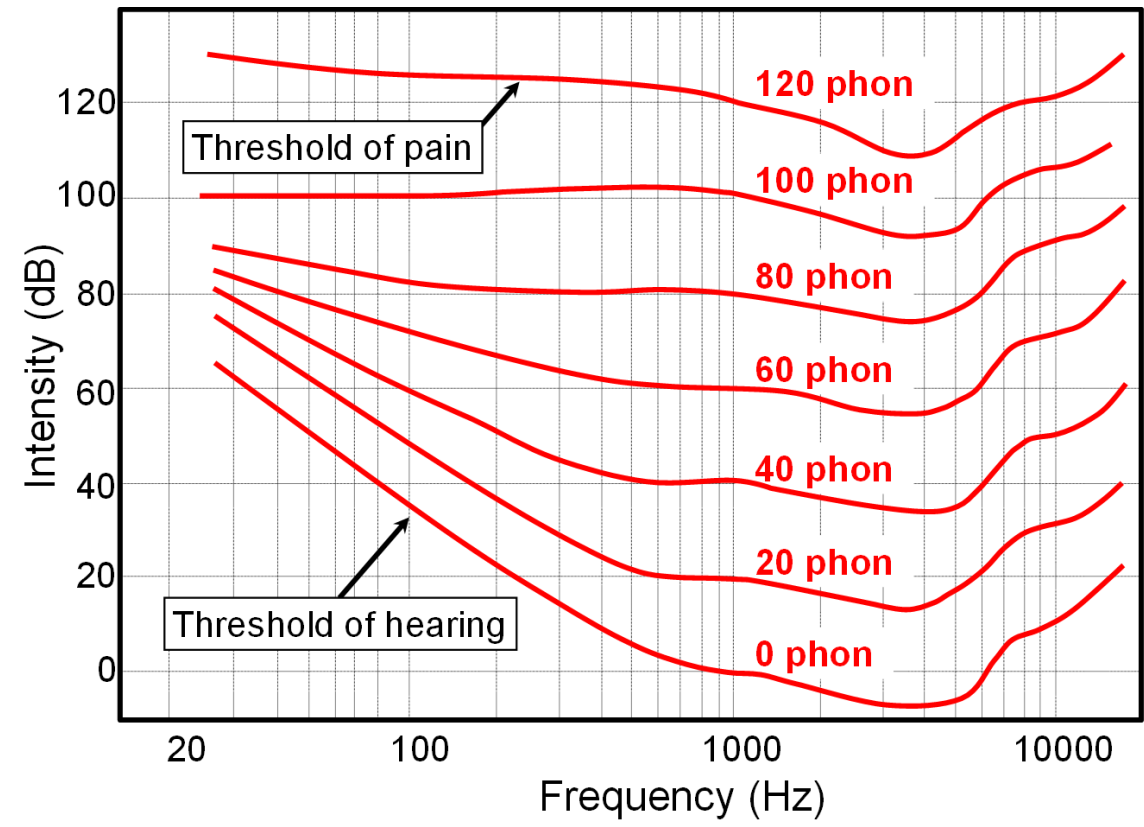


1.3 Audio Representation

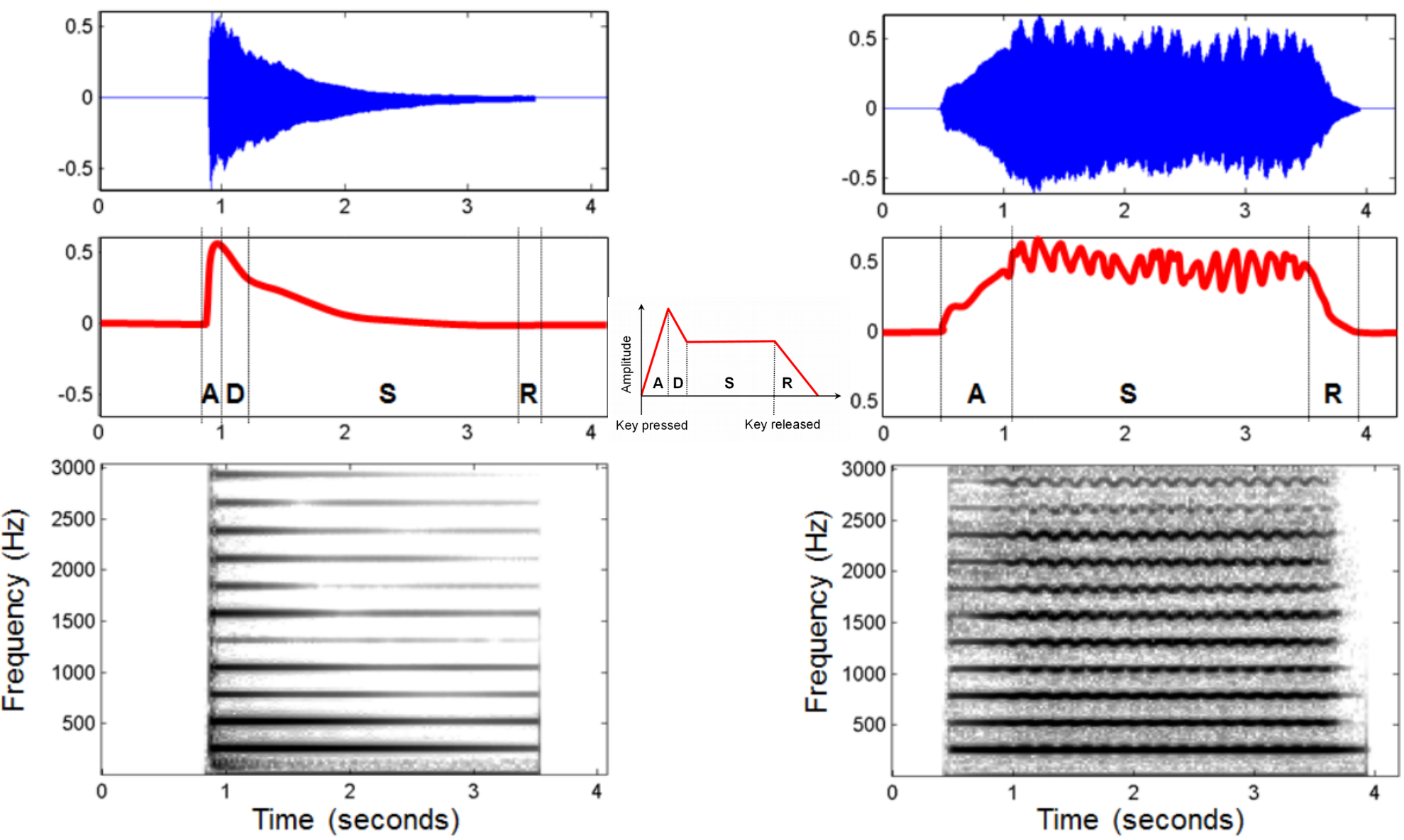


1.3 Audio Representation

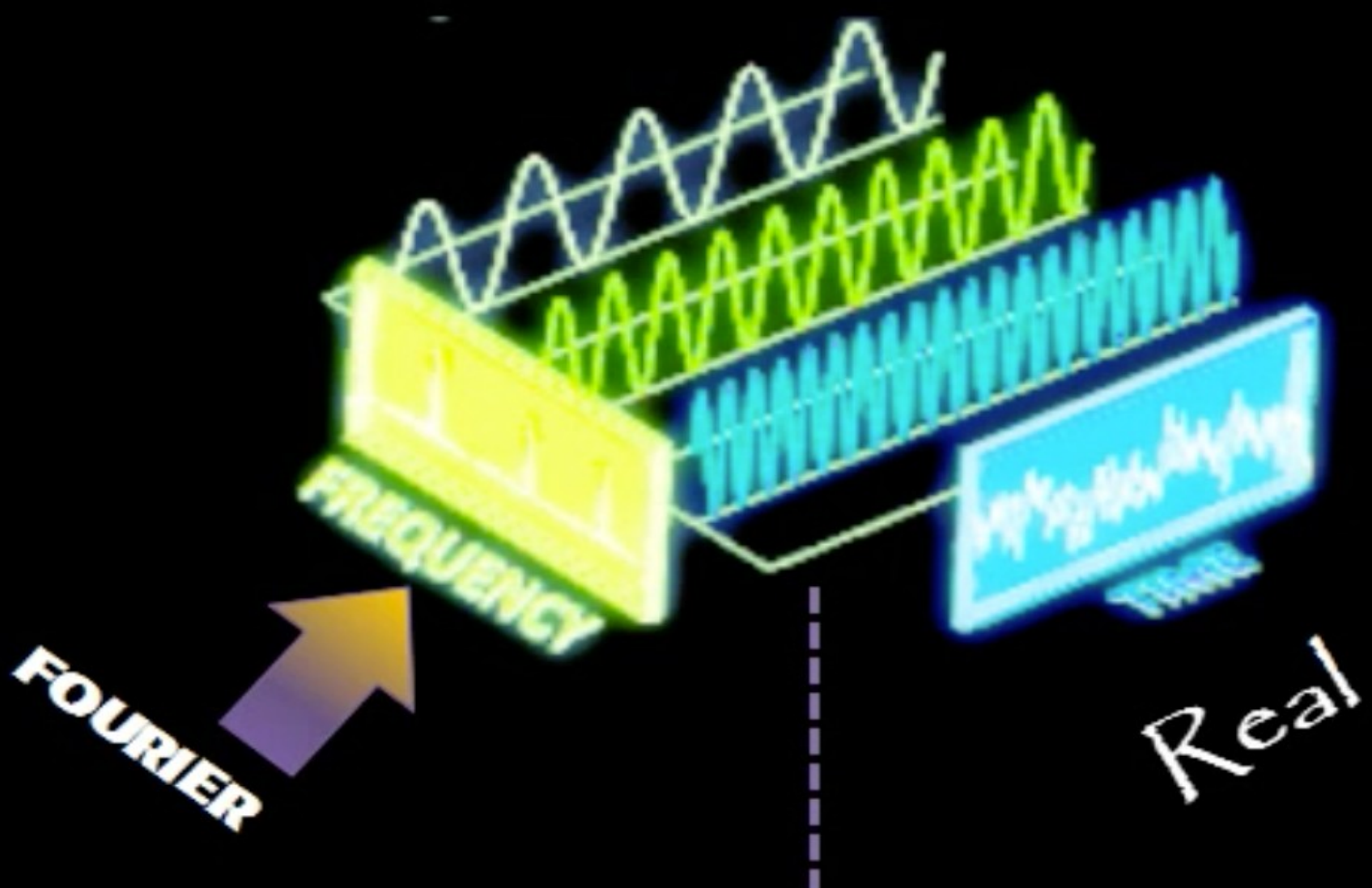
Source	Intensity	Intensity level	× TOH
Threshold of hearing (TOH)	10^{-12}	0 dB	1
Whisper	10^{-10}	20 dB	10^2
Pianissimo	10^{-8}	40 dB	10^4
Normal conversation	10^{-6}	60 dB	10^6
Fortissimo	10^{-2}	100 dB	10^{10}
Threshold of pain	10	130 dB	10^{13}
Jet take-off	10^2	140 dB	10^{14}
Instant perforation of eardrum	10^4	160 dB	10^{16}



1.3 Audio Representation



1.4 Fourier Analysis



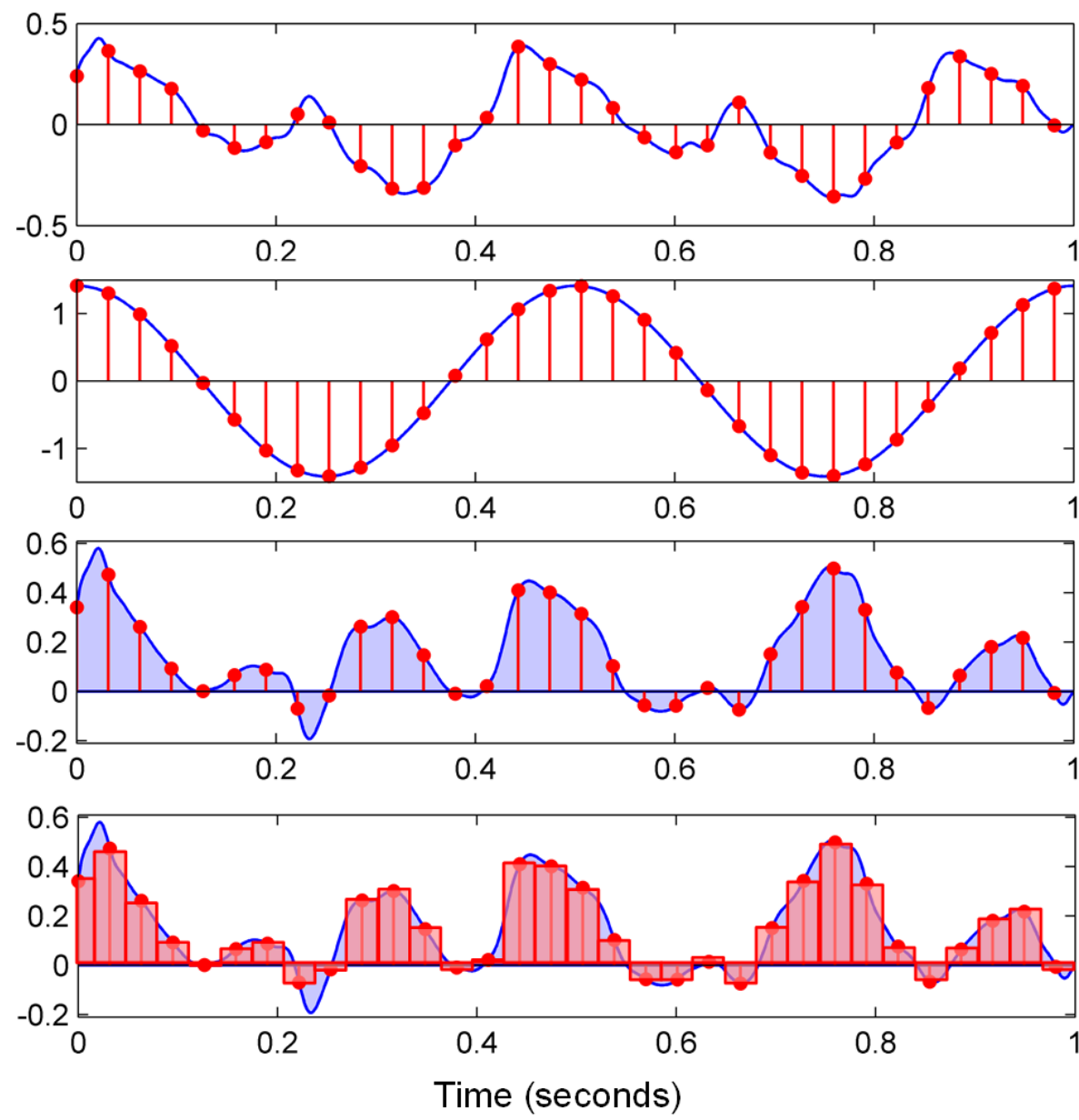
$$\sin\left(\frac{2\pi nx}{P} + \phi_n\right) \equiv \sin(\phi_n) \cos\left(\frac{2\pi nx}{P}\right) + \cos(\phi_n) \sin\left(\frac{2\pi nx}{P}\right)$$

$$\sin\left(\frac{2\pi nx}{P} + \phi_n\right) \equiv \operatorname{Re} \left\{ \frac{1}{i} \cdot e^{i\left(\frac{2\pi nx}{P} + \phi_n\right)} \right\} = \frac{1}{2i} \cdot e^{i\left(\frac{2\pi nx}{P} + \phi_n\right)} - \left(\frac{1}{2i} \cdot e^{-i\left(\frac{2\pi nx}{P} + \phi_n\right)} \right),$$

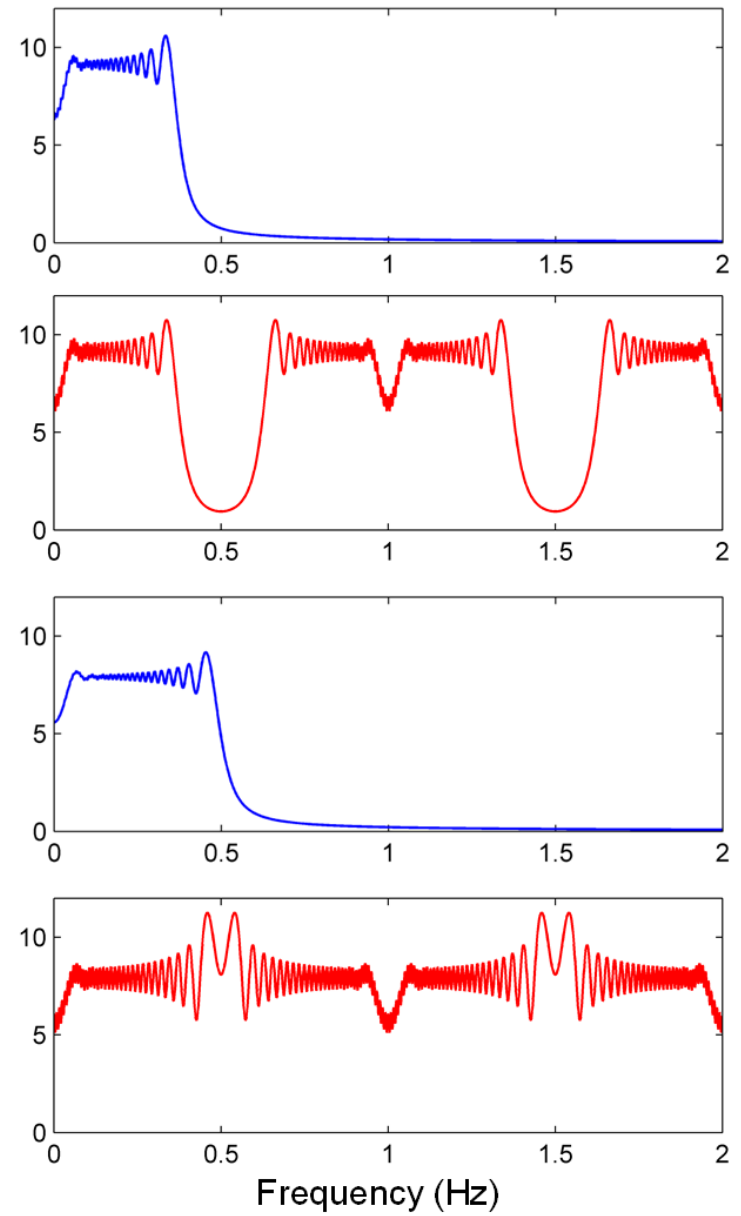
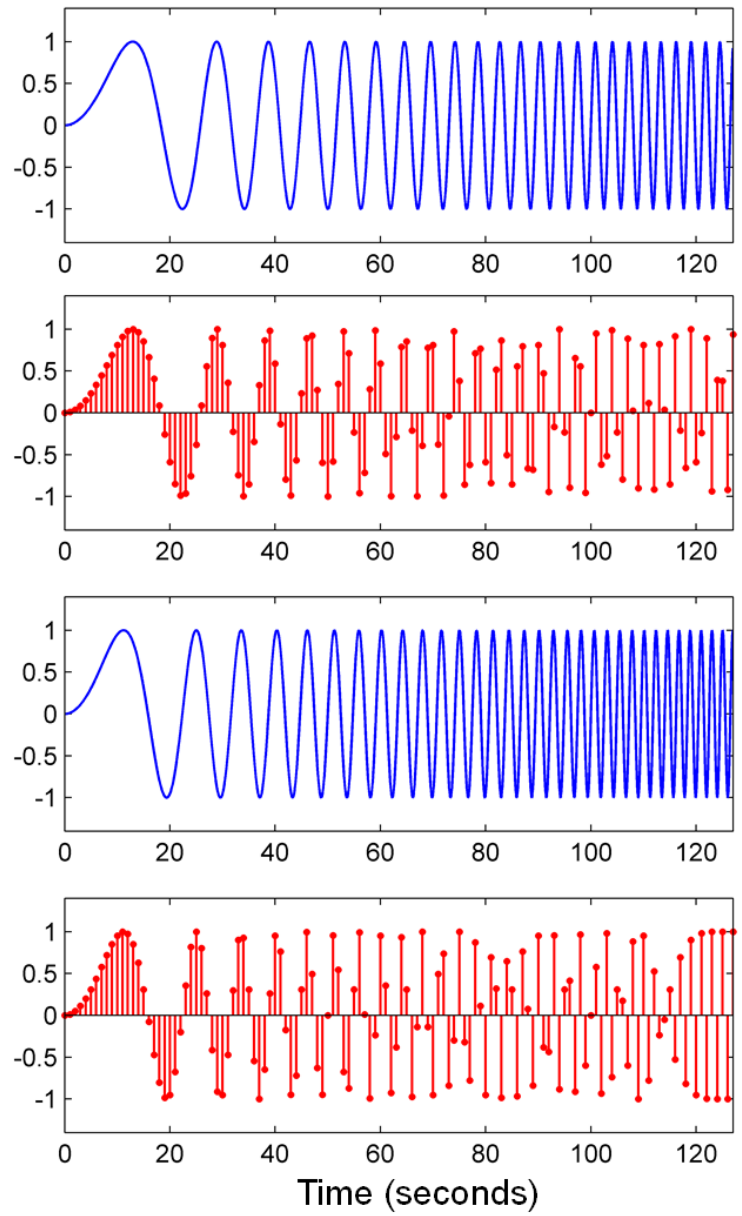
$$s_N(x) = \widehat{a_0} / 2 + \sum_{n=1}^N \left(\widehat{a_n} \cos\left(\frac{2\pi nx}{P}\right) + \widehat{b_n} \sin\left(\frac{2\pi nx}{P}\right) \right)$$

$$= \sum_{n=-N}^N c_n \cdot e^{i \frac{2\pi nx}{P}},$$

1.4 Fourier Analysis



1.4 Fourier Analysis



Time domain

Frequency domain



Input :

- Analogue:
Sound
Noise
Microphone
(AUX input)
- Digital:
Midi

Output:

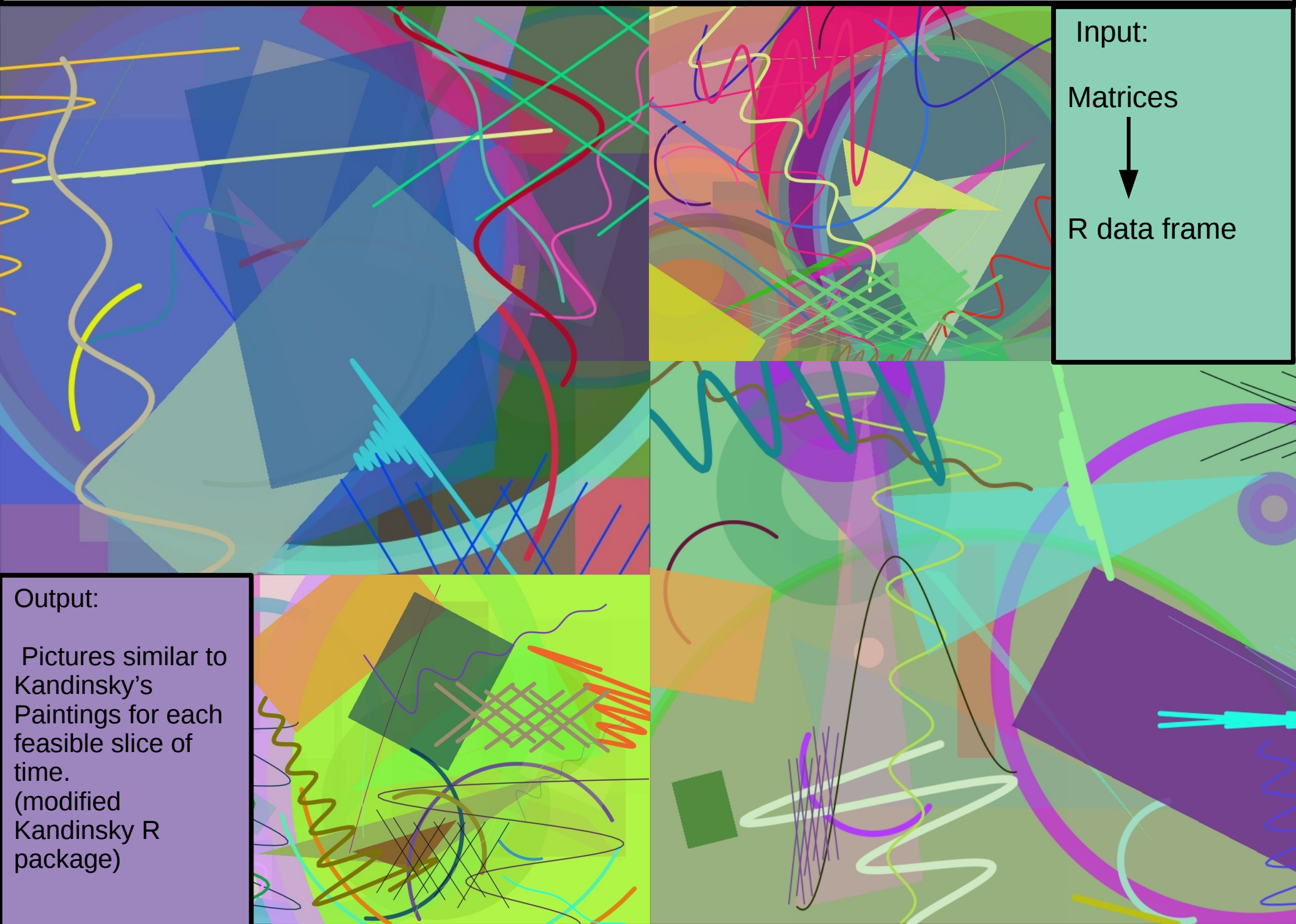
Matrices of numbers:

- Columns: Selected feature/Properties:
Pitch,ADSR,Velocity,Selected frequency,etc
- Rows: Slice of time considering beat or feasibility of CPU

How it is done:

With Processing , R , Python (Scripting Languages)

PART II : Visualization in R language



Input:
Matrices
↓
R data frame

Output:

Pictures similar to Kandinsky's Paintings for each feasible slice of time.
(modified Kandinsky R package)

Visualization in R language

```
require(grid)
randomKandinsky <- function(n = 10) {
  grid.newpage()

  grid.rect(gp=gpar(fill=rgb(runif(1),
    runif(1),
    runif(1),
    runif(1))))

  for (i in 1:n) {
    grid.rect(x = runif(1), y = runif(1), width = runif(1), height = runif(1),
      gp = gpar(col = NA,
        fill=rgb(runif(1),
          runif(1),
          runif(1),
          runif(1))))
    grid.circle(x = runif(1), y = runif(1), r = runif(1),
      gp = gpar(
        lwd = runif(1, 0, 100),
        col = rgb(runif(1),
          runif(1),
          runif(1)),
        fill=rgb(runif(1),
          runif(1),
          runif(1),
          runif(1))))
    grid.polygon(x = runif(3), y = runif(3),
      gp = gpar(col = NA,
        fill=rgb(runif(1),
          runif(1),
          runif(1),
          runif(1))))
    grid.curve(runif(1), runif(1), runif(1), runif(1),
      curvature = runif(1, -1, 1), square = FALSE, ncp = sample(100, 1),
      gp = gpar(lwd = runif(1, 0, 10),
        col = rgb(runif(1),
          runif(1),
          runif(1),
          1)))

    vp1 <- viewport(x = runif(1), y = runif(1), width = runif(1), height = runif(1), angle = runif(1) * 360)
    grid.rect(x = runif(1), y = runif(1), width = runif(1), height = runif(1),
      vp = vp1,
      gp = gpar(col = NA,
        fill=rgb(runif(1),
          runif(1),
          runif(1),
          runif(1))))

    vp2 <- viewport(x = runif(1), y = runif(1), width = runif(1), height = runif(1), angle = runif(1) * 360)
    gCurve(sin(x)/(x), sample(5, 1), sample(10:50, 1), vp = vp2,
      gp = gpar(lwd = runif(1, 0, 10),
        col = rgb(runif(1),
          runif(1),
          runif(1),
          1)))
  }
  vp3 <- viewport(x = runif(1), y = runif(1), width = runif(1), height = runif(1), clip = "off")
  gCrissCross(vp = vp3,
    gp = gpar(lwd = runif(1, 0, 10),
      col = rgb(runif(1),
        runif(1),
        runif(1),
        1)))
}
```

How it is done:

See the demo

devtools::install_github("nautilus69/kandinsky")

Sound file visualization with Fourier transformation in R

Descriptions: This Program Read a Wave music file from Local Hard Drive and Visualize the Fourier transformation processed music with Kandinsky-like output and animate them.

(I try to make a Web app with ShinyR so the user can interact with the parameters)

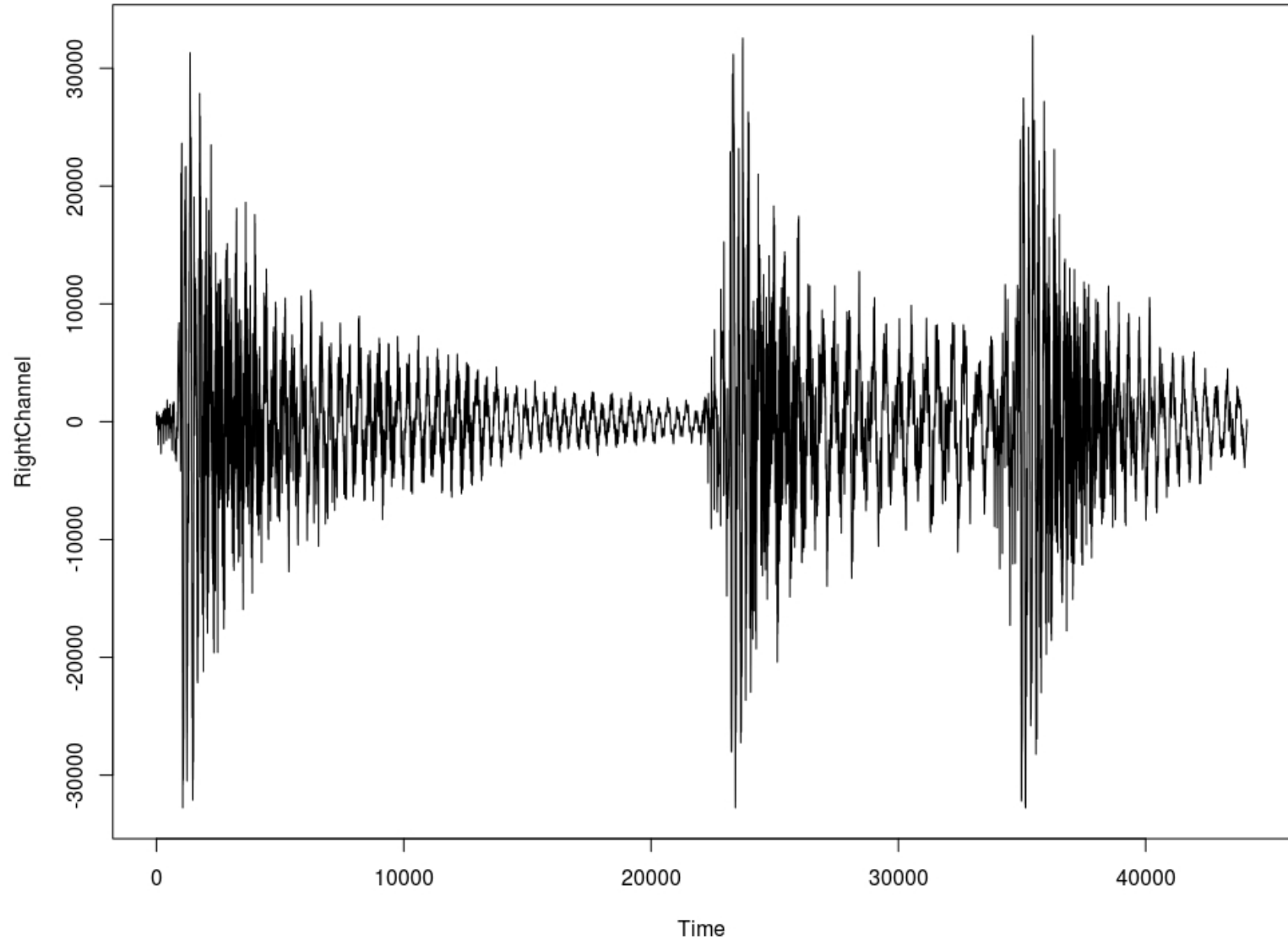
The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for reading a wave file, processing it, and generating a sequence of images for animation.
- Environment Pane:** Shows the current environment with a variable `myfourier1` of class `Wave`. It lists various values extracted from the wave file.
- Console:** Shows the execution progress, currently at line 58.

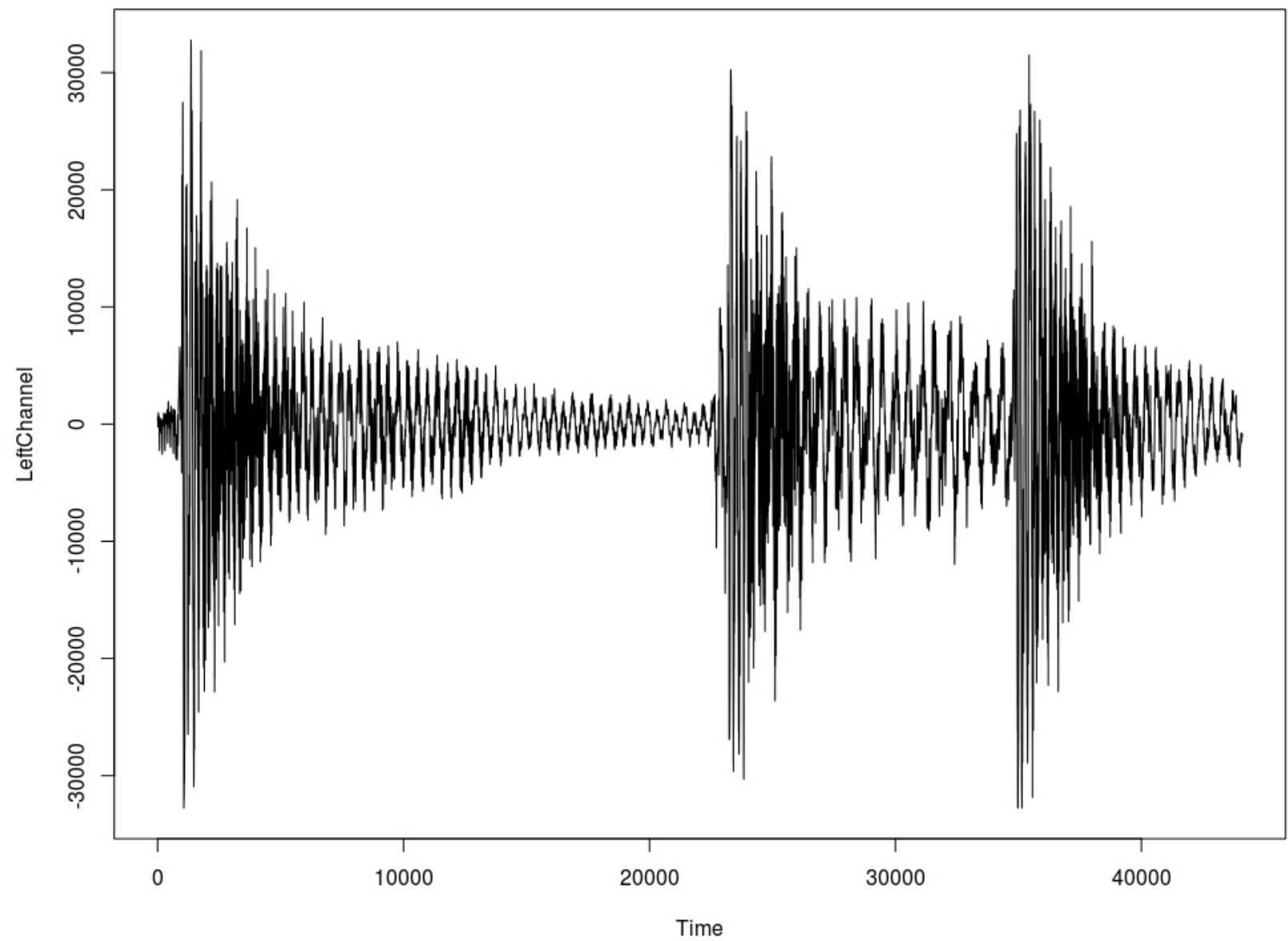
```
25 slider1 <- c(1,2)
26
27 start1 <- slider1[1]
28 end1 <- slider1[2]
29
30 ## Fourier Transformation of wave file
31 myfourier1 <- readWave(target_address, from = start1 , to = end1 , units = "seconds" )
32
33 #Parse Left Channel and Right Channel
34 LeftChannel <- myfourier1@left
35 RightChannel <- myfourier1@right
36
37 #Calculation of average of total channel amplitude
38 CenterChannel <- (RightChannel + LeftChannel)/2
39
40 # Plot the time series of fourier transformation of wave file and see if it's read correctly
41 ts.plot(CenterChannel)
42
43 # Define Indexer for crawling over the curve in the next loop
44 indexer1 <- seq( from = 1, to = 44100 , by = step1)
45 # adding end point to the indexer (seq() function doesnt have the extreme value)
46 indexer1 <- c(indexer1, 44100)
47
48 # Slice the Curve and use each batch(slice) for a unique kandinsky-like image
49 for(i in 1:(length(CenterChannel)/step1)){
50   KK<-CenterChannel[(indexer1[i]:indexer1[i+1])]
51   kandinsky(KK)
52   #save the file of each batch to address where we can finally animate them
53   dev.copy(jpeg, paste("/home/mahtab/Pictures/exports/",i,".jpg",sep = ""))
54   dev.off()
55 }
56
57 #animate the images in the folder using ffmpeg
58 system("cd /home/mahtab/Pictures/exports;ffmpeg -framerate 9 -pattern_type glob -i '*.jpg' -r 30 out4.mp4")
59
```

Global Environment	
myfourier1	Formal class Wave
Data	
CenterChannel	num [1:44100] 259 348 451 539 574 ...
end1	2
i	25L
indexer1	num [1:1471] 1 31 61 91 121 151 181 211 ...
KK	num [1:31] 270 214 143 45 -152 ...
LeftChannel	int [1:44100] 272 401 522 605 622 614 56...
RightChannel	int [1:44100] 246 296 380 473 525 566 56...
slider1	num [1:2] 1 2
start1	1
step1	30
target_address	"/home/mahtab/Music/test.wav"

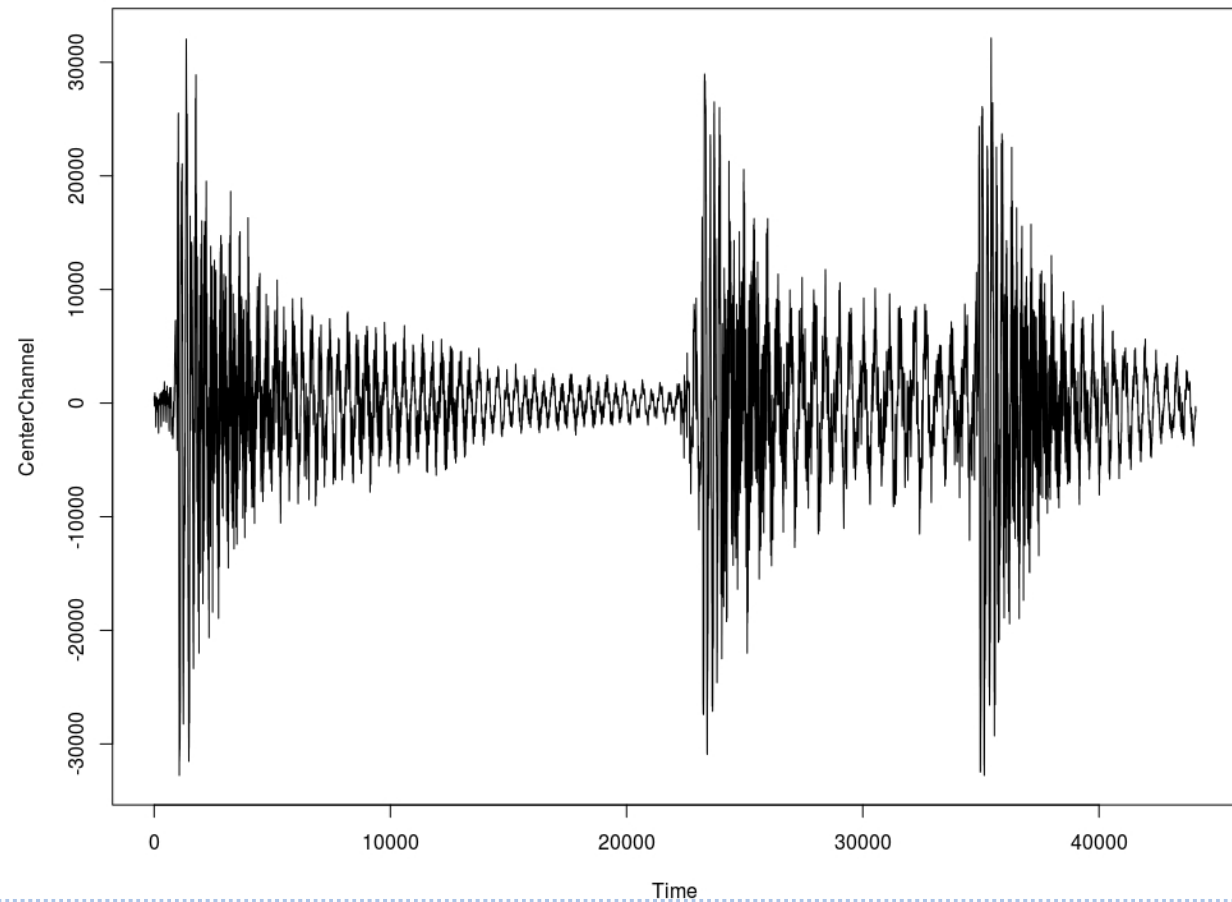
Sound file visualization with Fourier transformation in R



Sound file visualization with Fourier transformation in R



Sound file visualization with Fourier transformation (Amplitude per time)



Calculation of average of total channel amplitude
`CenterChannel <- (RightChannel + LeftChannel)/2`

Plot the time series of fourier transformation of wave file and see if it's read correctly
`ts.plot(CenterChannel)`

DEMO

Thank You