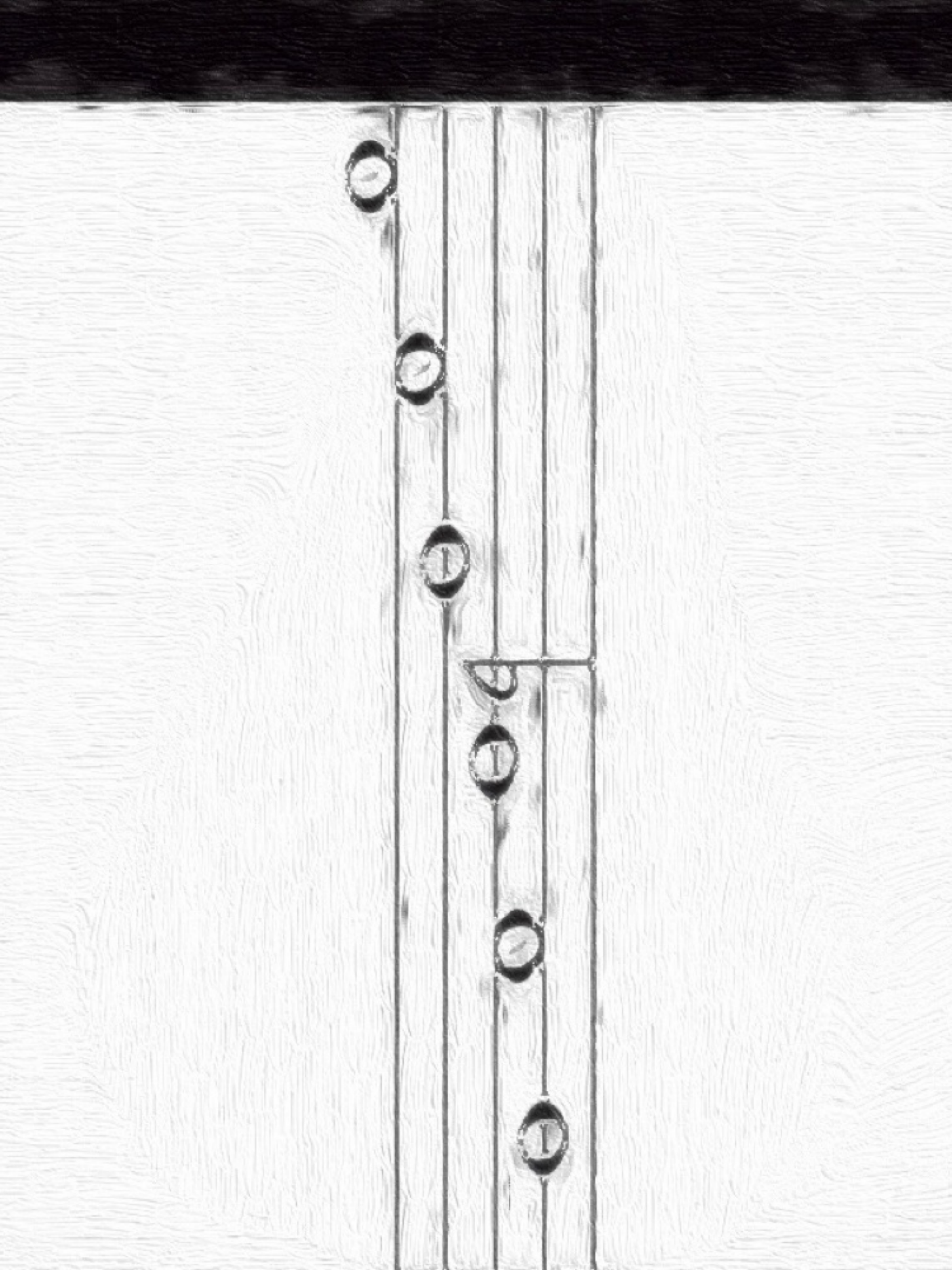
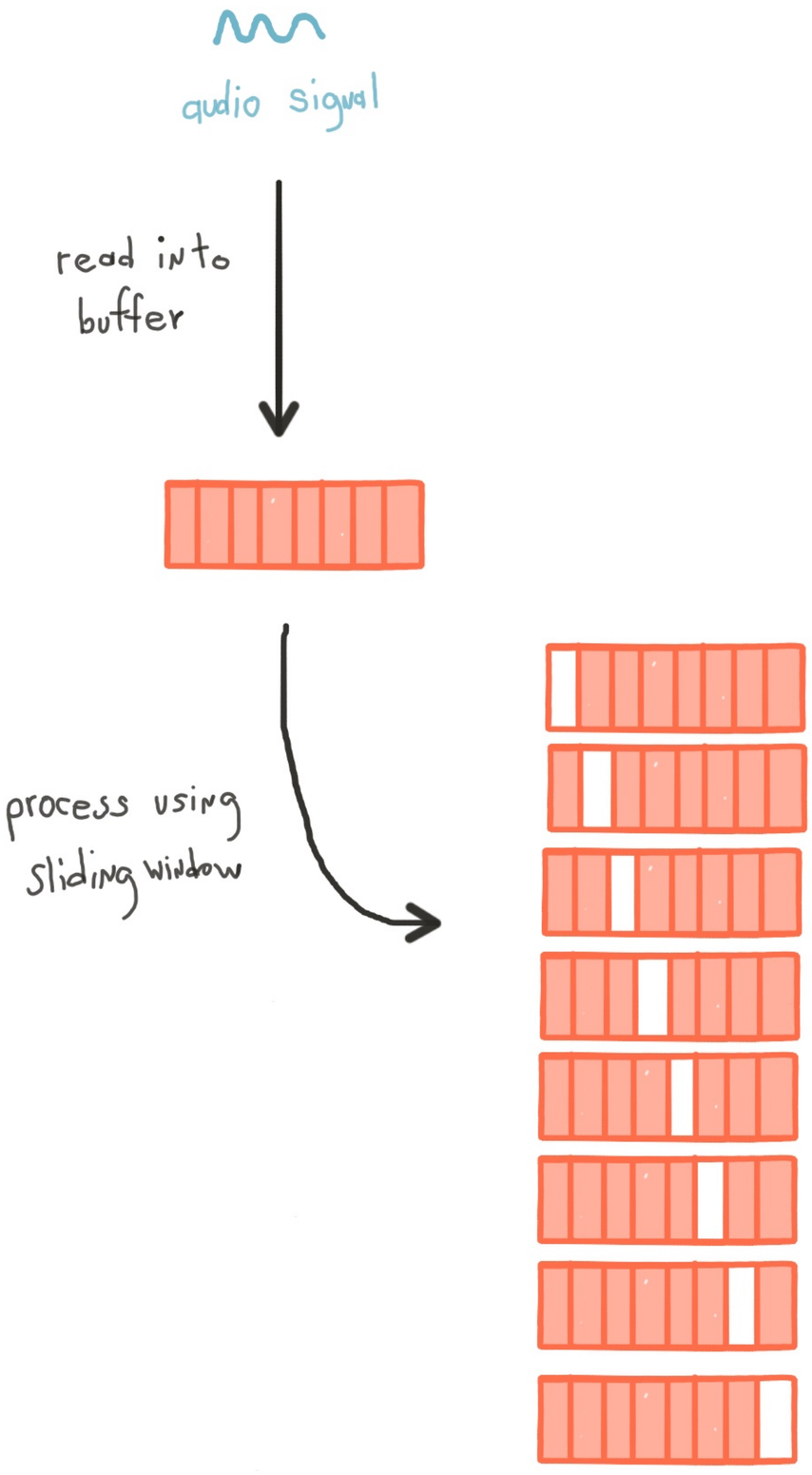



audio signal
read into buffer
process using sliding window

packing for fft result = $\{[DC, NY], C[1], C[2], \ldots, C[N/2]\}$

vDSP_fft_zrip:

$$C_m = 2\,FDFT(X_m)$$

$$FDFT(X_m) = \sum_{n=0}^{N-1} x_n e^{\frac{(-j2\pi nm)}{N}}$$

v_DSP_zvmags: $C_{n|C} = (Re[A_{n|A}])^2 + (Im[A_{n|A}])^2$ for $n = \{0, N-1\}$

$Spec(m,k) = v$

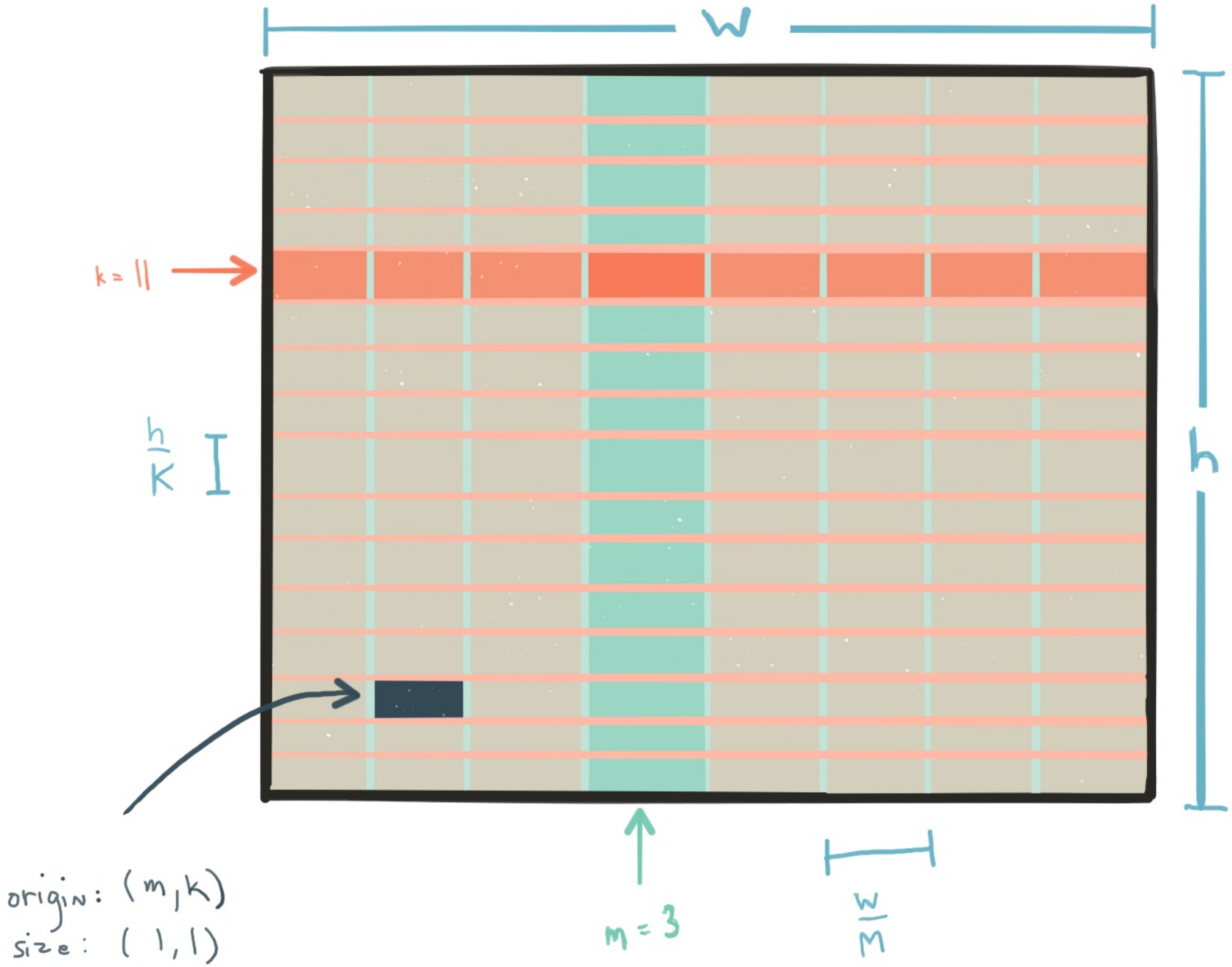
$m \in [0 : M-1]$

$k \in [0 : K-1]$

N = window size

M = total frames

$K = \frac{N}{2}$



Linear Y (frequency) Axis

F_s	sample rate
$w(l)\ \ l \in [0:N-1]$	Window function's nonzero values
$k \in [0:K]$	Coefficient index
$K = \frac{N}{2}$	Nyquist frequency
$n \in [0:M-1]$	STFT frame
$p \in [0:127]$	MIDI pitch

$$F_{coef}(k) = \frac{k F_s}{N}$$

$$F_{pitch}(p) = 2^{\frac{(p-69)}{12}} \cdot 440$$

$$P_{freq}(f) = 12 \log_2\left(\frac{f}{440}\right) + 69$$

let $f = F_{pitch}(p)$

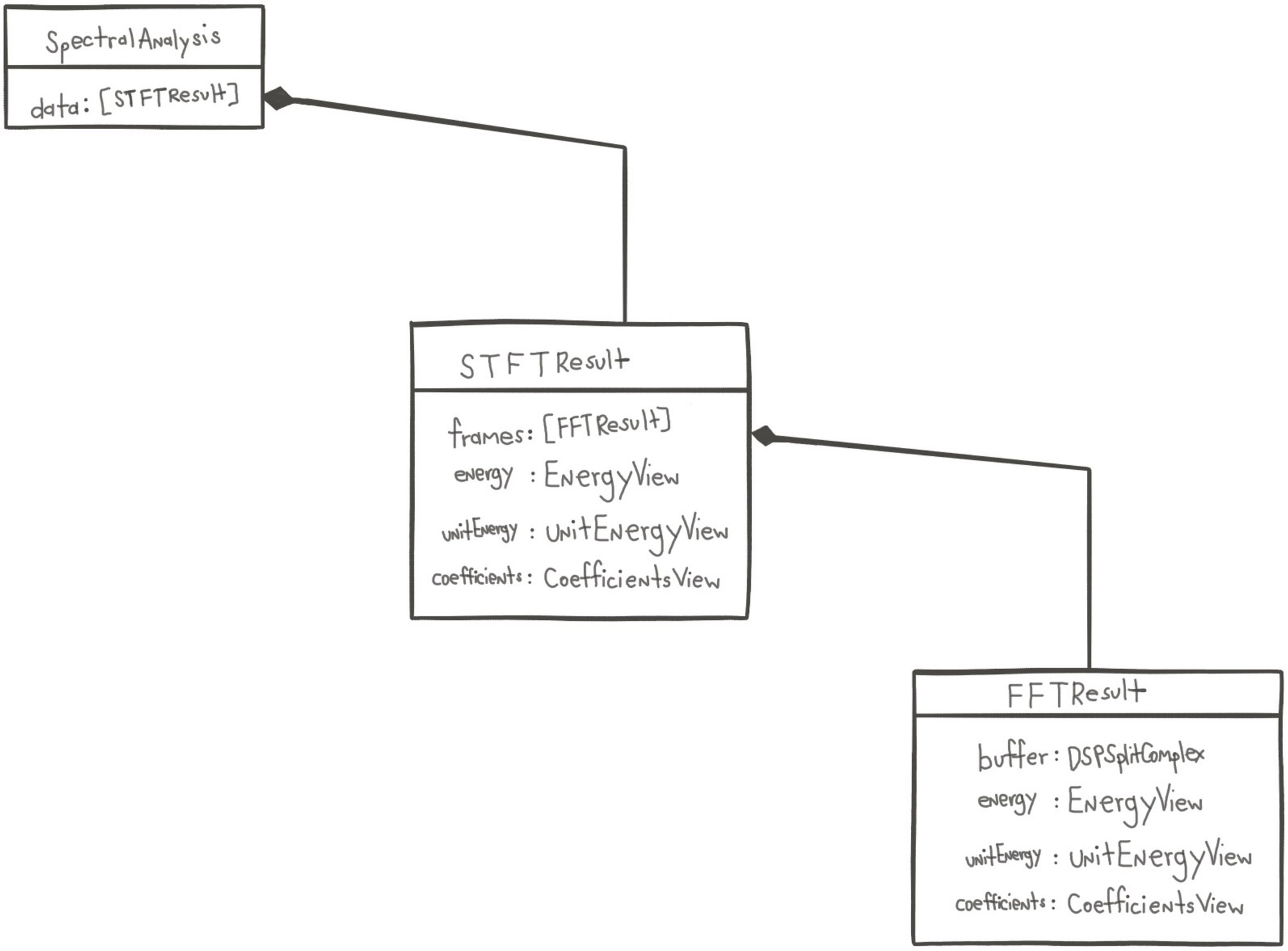
then $f = 2^{\frac{(p-69)}{12}} \cdot 440$

$$\frac{f}{440} = 2^{\frac{(p-69)}{12}}$$

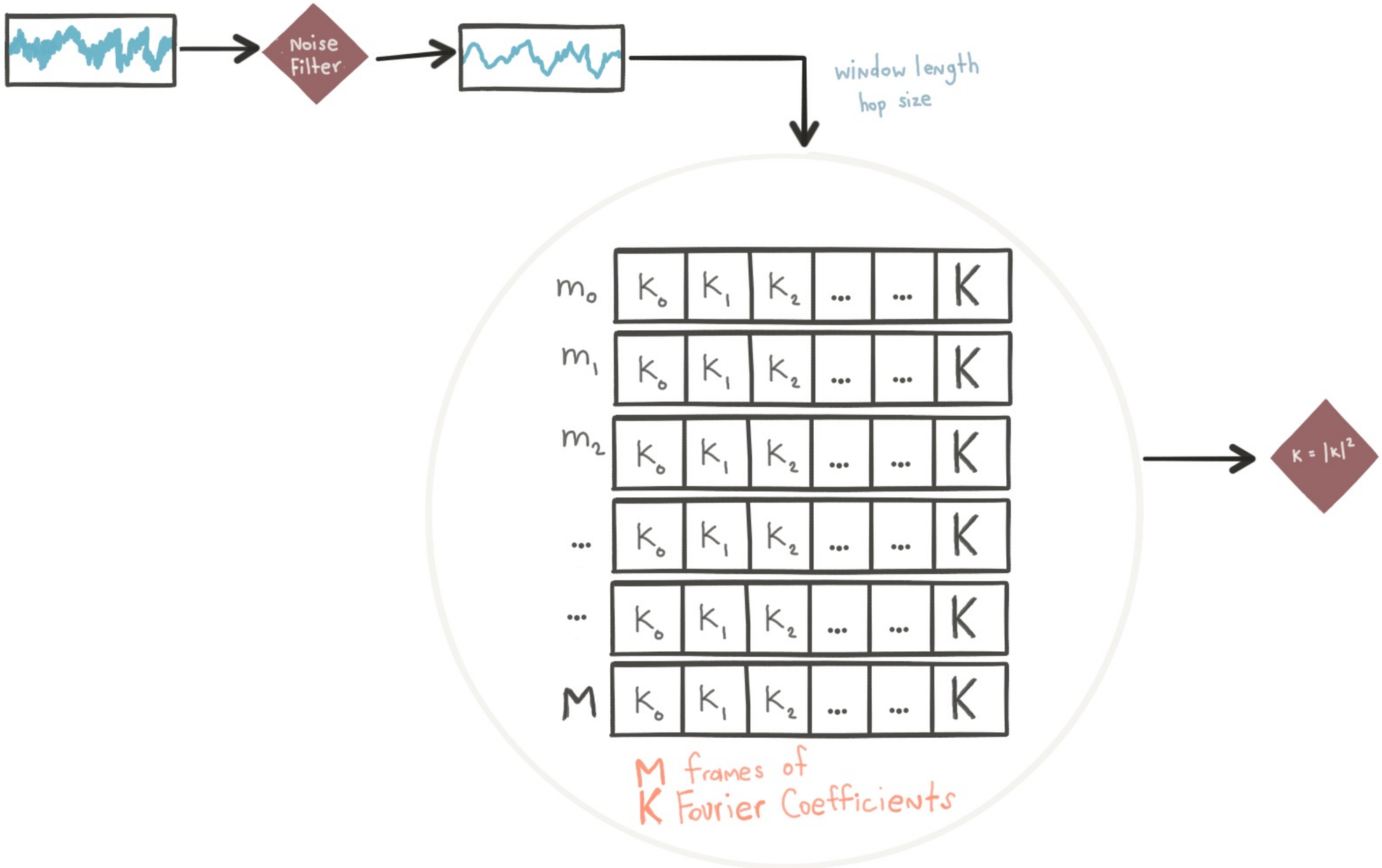
$$\log_2\left(\frac{f}{440}\right) = \frac{p-69}{12}$$

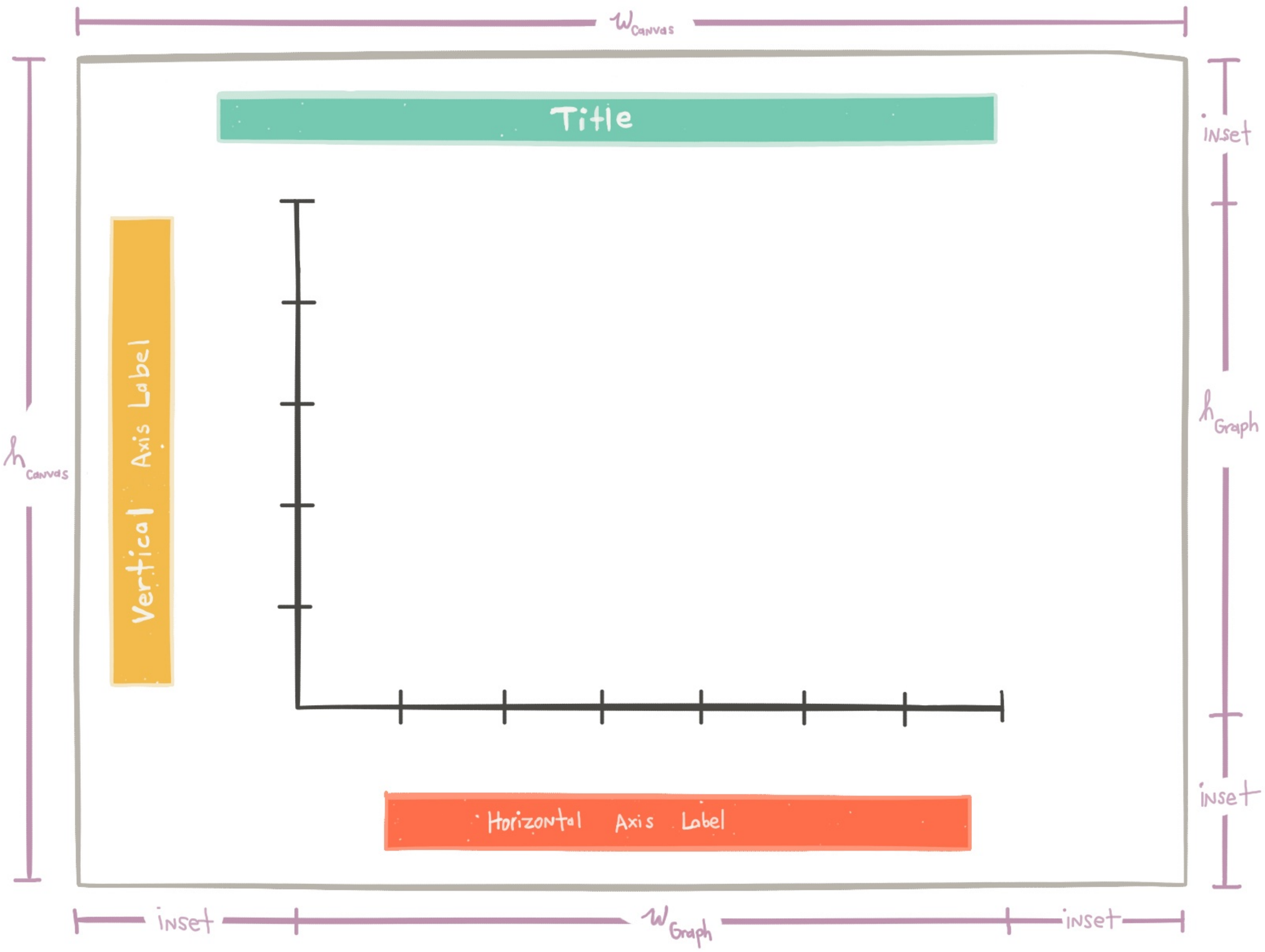
$$12 \log_2\left(\frac{f}{440}\right) = p - 69$$

$$12 \log_2\left(\frac{f}{440}\right) + 69 = p$$

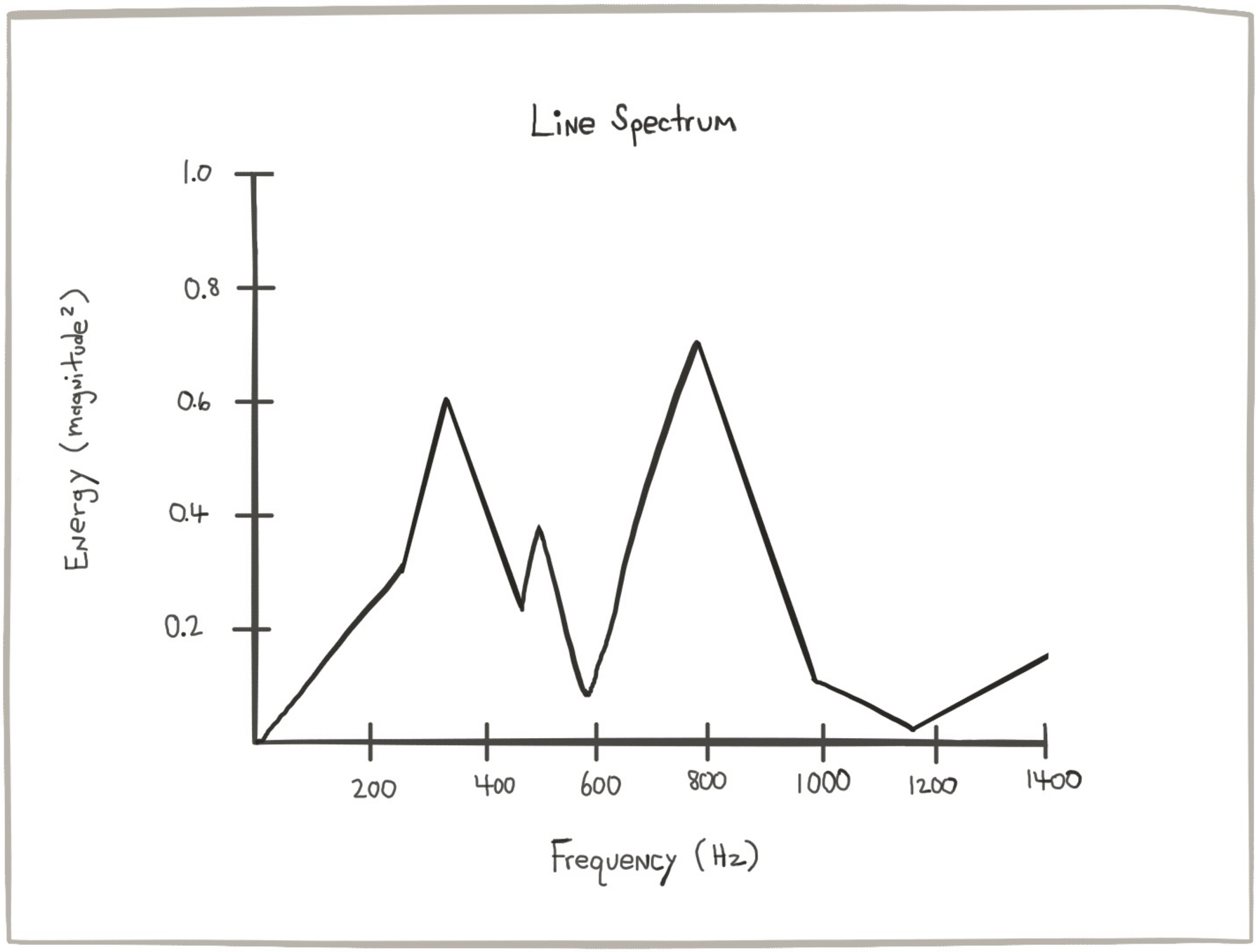

SpectralAnalysis
data: [STFTResult]
STFTResult
frames: [FFTResult]
energy : EnergyView
unitEnergy : UnitEnergyView
coefficients : CoefficientsView
FFTResult
buffer : DSPSplitComplex
energy : EnergyView
unitEnergy : UnitEnergyView
coefficients : CoefficientsView

Fill buffers until samples amount to a specific length of time.


Noise Filter
window length
hop size
m_0 K_0 K_1 K_2 ... ... K
m_1 K_0 K_1 K_2 ... ... K
m_2 K_0 K_1 K_2 ... ... K
... K_0 K_1 K_2 ... ... K
... K_0 K_1 K_2 ... ... K
M K_0 K_1 K_2 ... ... K
M frames of
K Fourier Coefficients
$K = |k|^2$

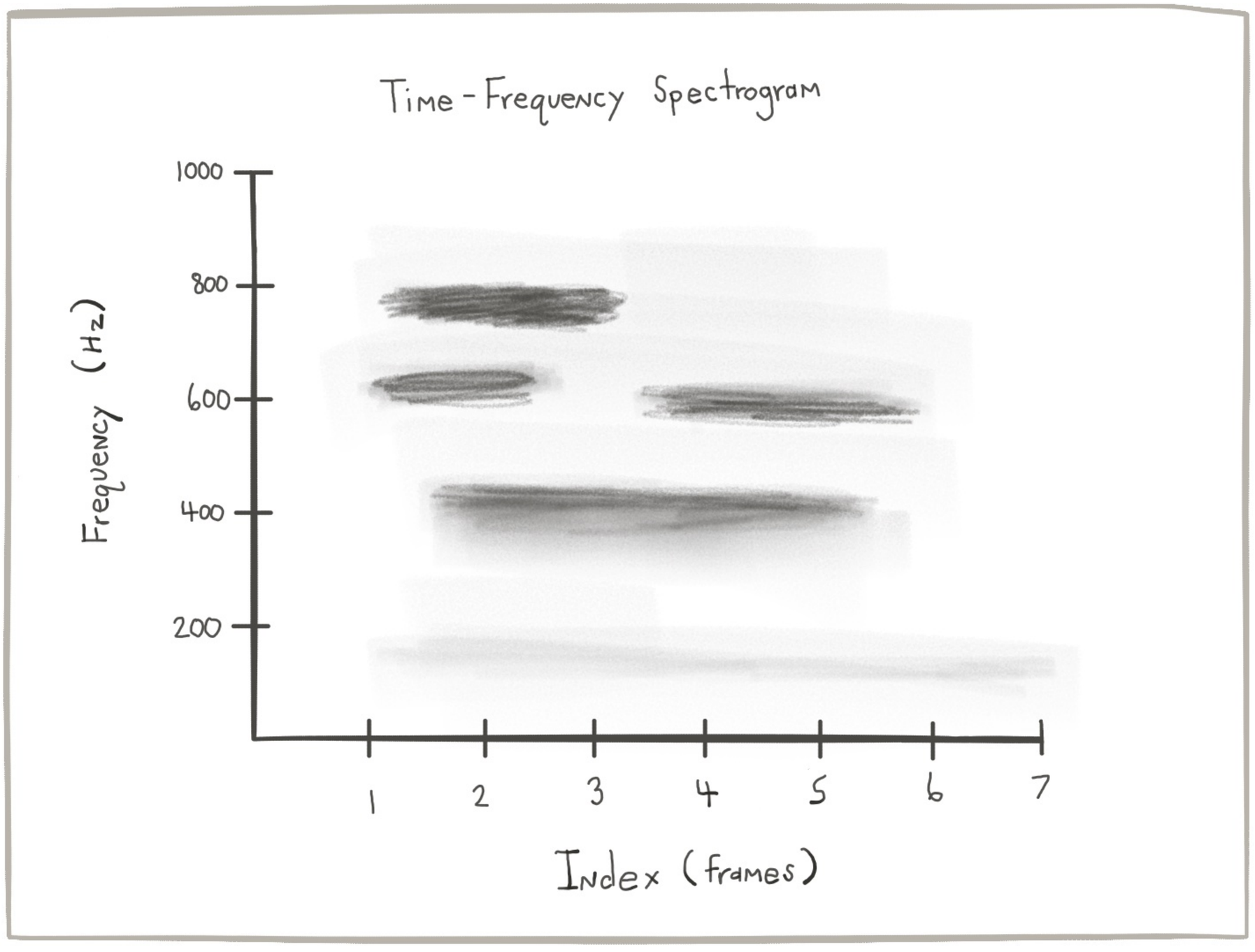

w_{Canvas}
Title
Vertical Axis Label
Horizontal Axis Label
h_{Canvas}
inset
h_{Graph}
inset
inset
w_{Graph}
inset

$$E_{coef}(k) = |x(k)|^2$$

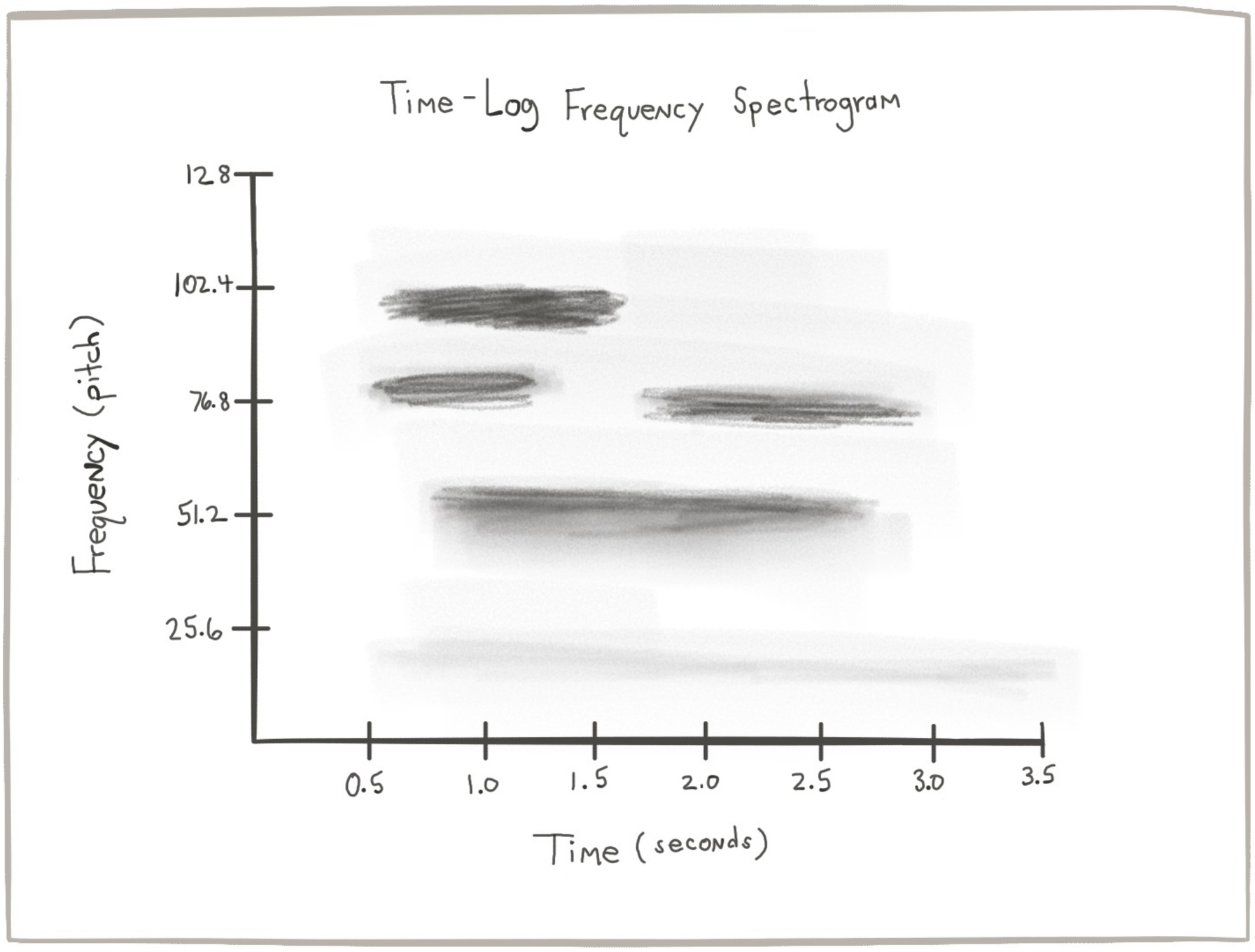


$$F_{coef}(k) = \frac{k \cdot F_s}{N}$$

$$F_{coef}(k) = \frac{k \cdot F_s}{N}$$



$$P_{coef}(k) = 12 \log_2\left(\frac{k \cdot F_s}{440 \cdot N}\right) + 69$$



$$T_{coef}(m) = \frac{m \cdot H}{F_s}$$

$$F_{pitch}(p) = 2^{\frac{p-69}{12}} \cdot 440$$

$$\frac{F_{pitch}(p)}{440} = 2^{\frac{p-69}{12}}$$

$$\log_2\left(\frac{F_{pitch}(p)}{440}\right) = \frac{p-69}{12}$$

$$p = 12\log_2\left(\frac{F_{pitch}(p)}{440}\right) + 69$$

$$F_{coef}(k) = \frac{k \cdot F_s}{N}$$

$$F_{pitch}(p) = F_{coef}(k)$$

$$P_{coef}(k) = 12\log_2\left(\frac{k \cdot F_s}{440 \cdot N}\right) + 69$$

L_{axis} = the length of the axis in points

L_{values} = the diameter of the range of values in unspecified units

$$\frac{1}{L_{axis}} = \frac{x}{L_{values}}$$

$$x = \frac{L_{values}}{L_{axis}} = \text{ratio of unspecified value units per point}$$

$$\frac{y}{L_{axis}} = \frac{1}{L_{values}}$$

$$y = \frac{L_{axis}}{L_{values}} = \text{ratio of points per unspecified value unit}$$

$$y \cdot n = 200$$

$$n = \left\lceil \frac{200}{y} \right\rceil$$

given unit values of $1\,Hz$, $L_{axis} = 1600\,pt$, $L_{values} = 22{,}039.234375\,Hz$

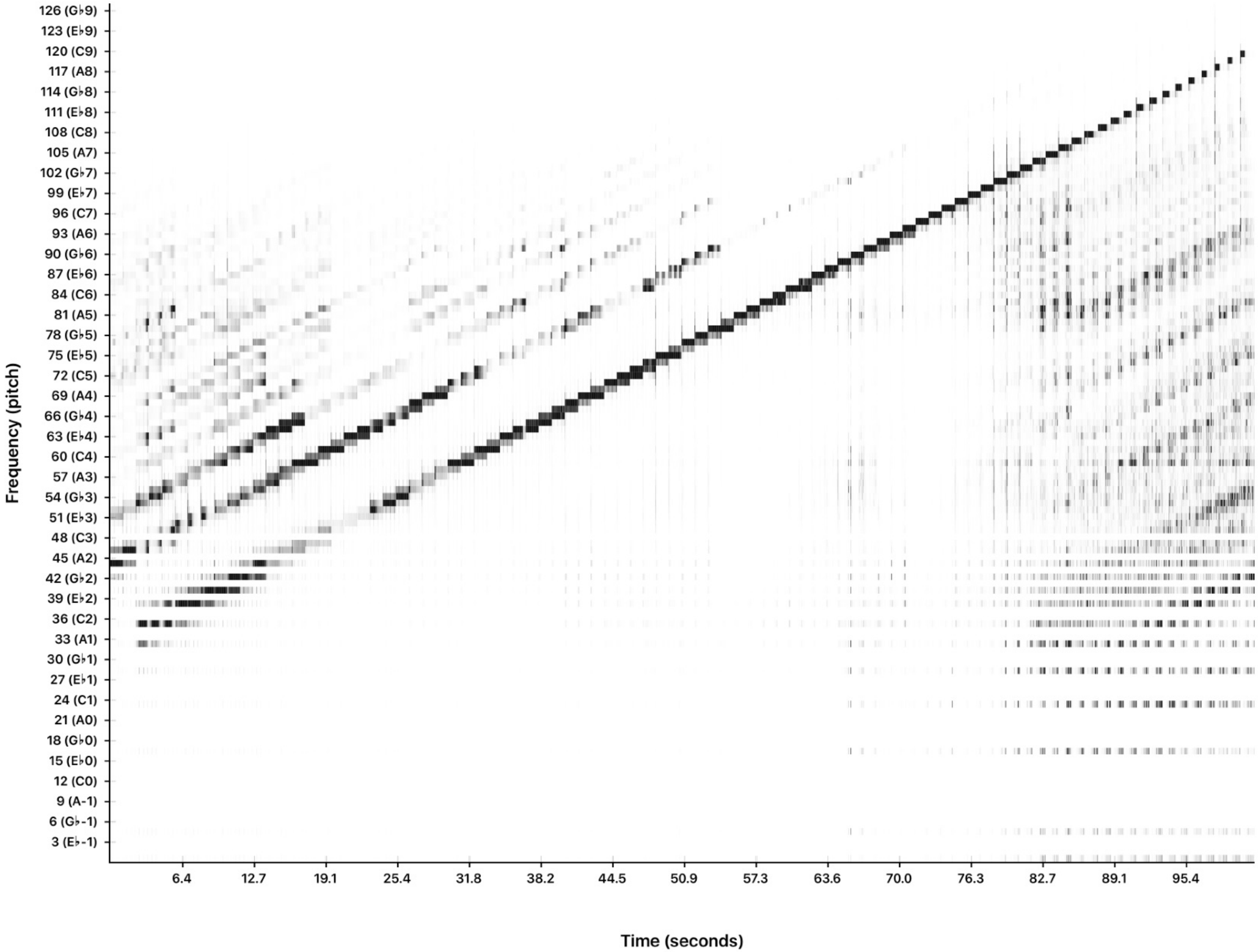
$$x = \frac{22{,}039.234375\,Hz}{1600\,pt} = 13.77452148\,Hz/pt$$

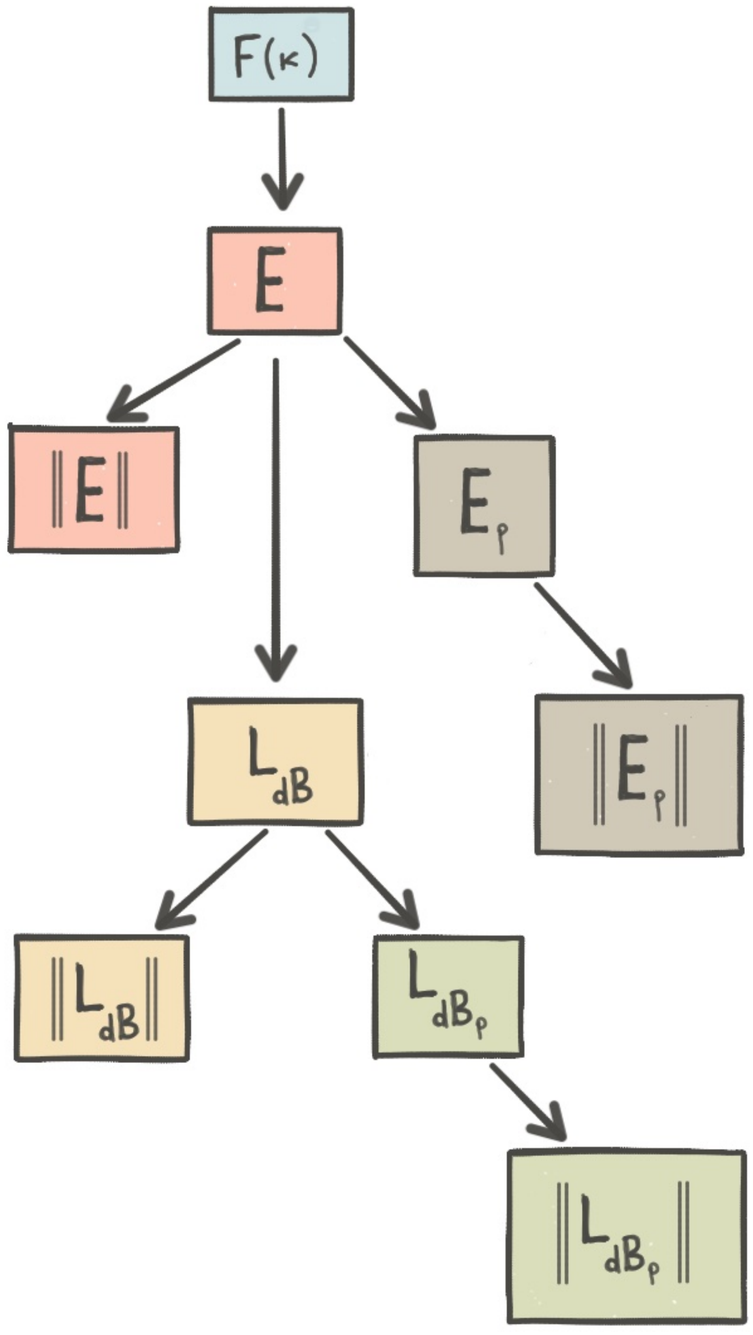
$$y = \frac{1600\,pt}{22{,}039.234375\,Hz} = 0.07259780321\,pt/Hz$$

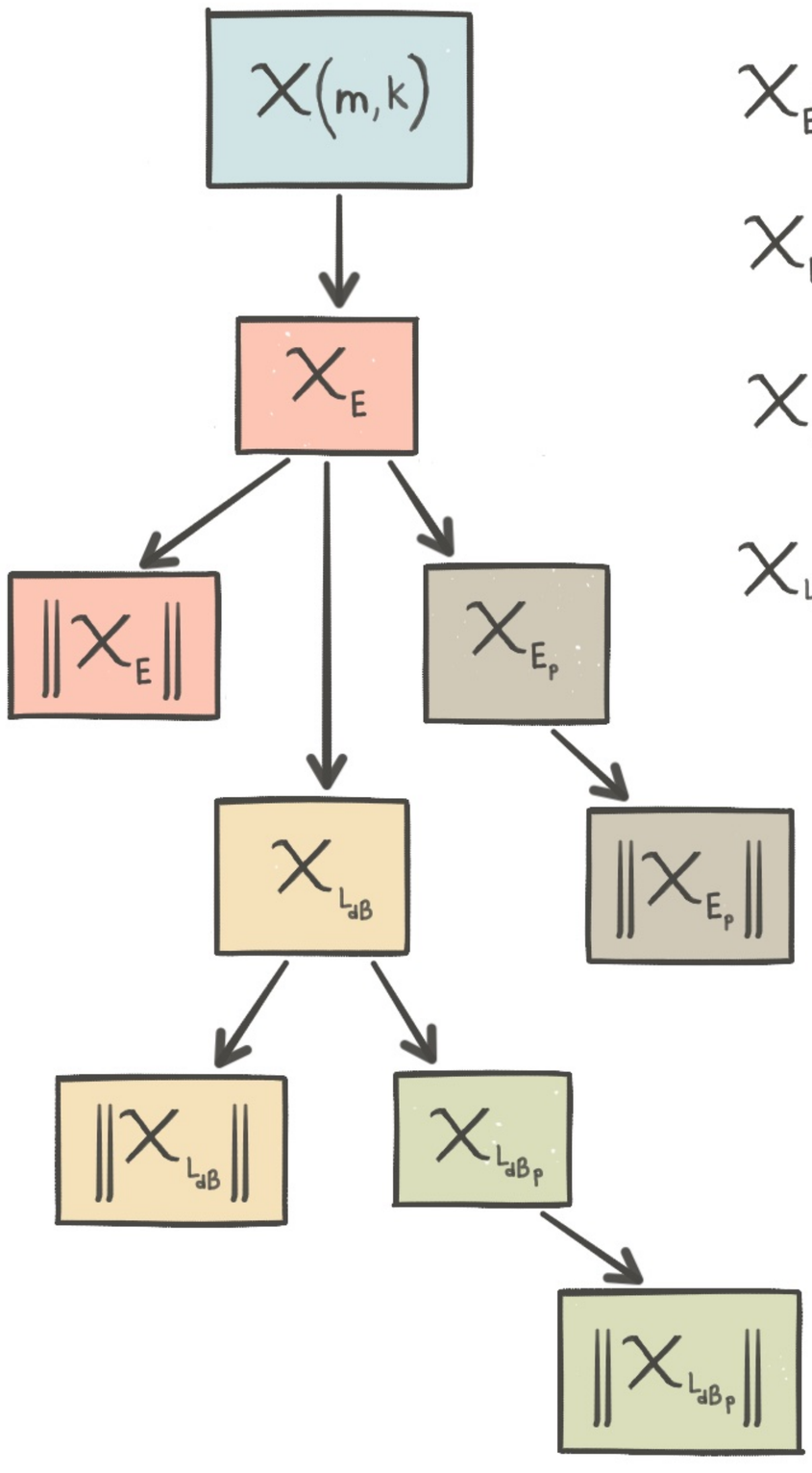
$$\frac{200\,pt}{0.07259780321\,pt/Hz} = \frac{200\,pt \cdot 1\,Hz}{0.07259780321\,pt} = 2754.904297\,Hz$$

$$2{,}755\,Hz \cdot 0.07259780321\,pt/Hz = 200.0069478\,pt$$

Log-Frequency Spectrogram of Chromatic_Scale-Steinway_Grand.aif







$$X_E = X(m,k) = \sum_0^{m-1} E$$

$$X_{E_\rho} = X(m,k) = \sum_0^{m-1} E_\rho$$

$$X_{L_{dB}} = X(m,k) = \sum_0^{m-1} L_{dB}$$

$$X_{L_{dB_\rho}} = X(m,k) = \sum_0^{m-1} L_{dB_\rho}$$

$$E = \left| F(k) \right|^2$$

$$E_\rho = \sum_{k \in P(\rho)} E \quad \text{where } \rho \in [0:127]$$

$$L_{dB} = 21 \log_{10} E$$

$$L_{dB_\rho} = \sum_{k \in P(\rho)} L_{dB} \quad \text{where } \rho \in [0:127]$$

$$\|f(x)\| = f(x) \rightarrow [0:1]$$

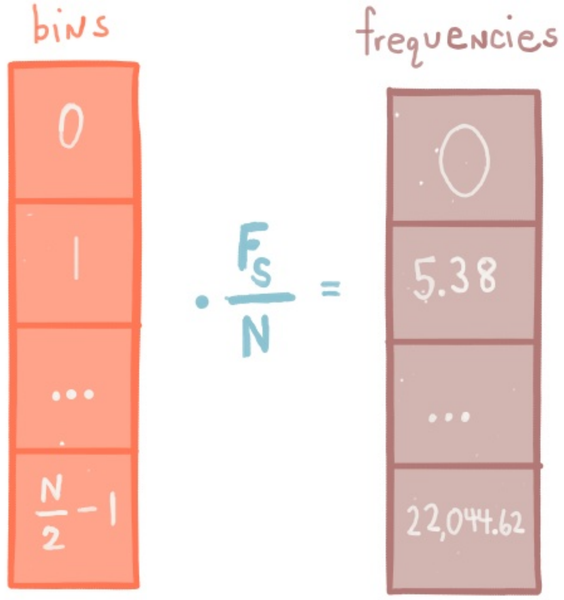
$$F_{pitch}(p) = 2^{\frac{p-69}{12}} \cdot 440$$

$$\frac{F_{pitch}(p)}{440} = 2^{\frac{p-69}{12}}$$

$$\log_2\left(\frac{F_{pitch}(p)}{440}\right) = \frac{p-69}{12}$$

$$p = 12\log_2\left(\frac{F_{pitch}(p)}{440}\right) + 69$$

$$F_{coef}(k) = \frac{k \cdot F_s}{N}$$



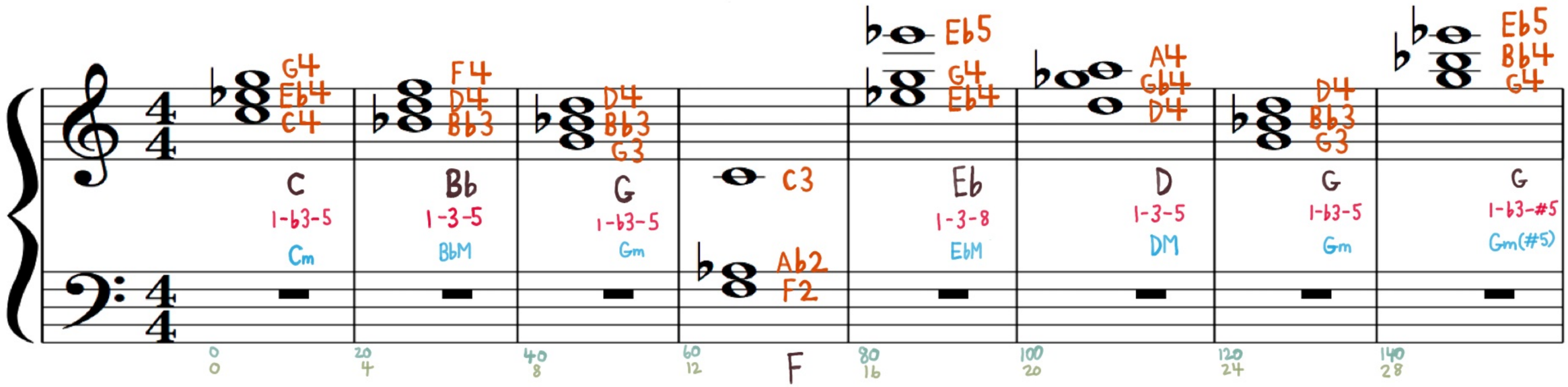
let $w = 8192$, $h = 4096$, $F_s = 44100$

$$F_{pitch}(p) = F_{coef}(k)$$

$$2^{-\frac{68}{12}}$$

$$p_{coef}(k) = 12\log_2\left(\frac{k \cdot F_s}{440 \cdot N}\right) + 69$$

Note Root Intervals Chord



Frame (feature rate = 10 Hz) Frame (feature rate = 2 Hz)

Chroma : Int

c=0, db, d, eb, e, f, gb, g, ab, a, bb, b=11

Pitch

rawValue : Int ← 0-127 = MIDI notes

centerFrequency : Frequency ← $2^{\left(\frac{r-69}{12}\right)} \cdot 440$

toneHeight : Int ← $\frac{r}{12} - 1$

chroma : Chroma ← r mod 12

Tone

fundamental : Frequency

subscript[harmonic:UInt8] → Frequency ← fundamental · harmonic

Frequency

rawValue : Float ← Hz

pitch : Pitch ← $12 \log_2\left(\frac{r}{440}\right) + 69$

Rational Transfer Function

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(n_b+1)z^{-n_b}}{1 + a(2)z^{-1} + \ldots + a(n_a+1)z^{-n_a}} X(z)$$

where n_a = feedback filter order

n_b = feedforward filter order

or

$$a(1)y(n) = b(1)x(n) + b(2)x(n-1) + \ldots + b(n_b+1)x(n-n_b) - a(2)y(n-1) - \ldots - a(n_a+1)y(n-n_a)$$

$$a_0 y_n = b_0 x_n + b_1 x_{n-1} + \ldots + b_8 x_{n-8} - a_1 y_{n-1} - \ldots - a_8 y_{n-8}$$

filterCoefficients [60]

	a
0	1.0
1	-7.370440011388690 3
2	24.356331306218301
3	-47.053025057578601
4	58.073443623397
5	-46.877802066188003
6	24.175265864191601
7	-7.288405125671089 6
8	0.985187314503690 1

	b
0	0.003143643799764059 9
1	-0.023195340053373301
2	0.076751499507004203
3	-0.148500007304070 01
4	0.183602168182193 99
5	-0.148500007304070 01
6	0.076751499507004398
7	-0.023195340053373401
8	0.003143643799764069 9

	x
0	0.0
1	-3.05176e-05
2	-6.10352e-05
3	-6.10352e-05
4	-0.00012207
5	-9.15527e-05
6	-0.000152588
7	-0.000152588
8	-0.000152588
9	-0.000213623
10	-0.000213623
11	-0.000183105
12	-0.000244141
13	-0.000152588
14	-0.000244141
15	-0.000152588
16	-0.000213623
17	-0.000152588
18	-0.000122207
19	-0.000122207
20	3.05176e-05
21	-3.05176e-05
22	9.15527e-05
23	9.15527e-05
24	0.000152588
25	0.000244141
26	0.000305176
27	0.000366211
28	0.000396729
29	0.000396729
30	0.000457764
31	0.000457764
32	0.000427246
33	0.000488281

$y_0 = 0$

$y_1 = -9.593646402e-08 + 0 - 0$

MatLab ChromaToolbox Routine

Demo

convert audio file
estimate tuning ← ignore for now
STMSP window length = 4410
convert audio to pitch using filter bank

wav_to_audio

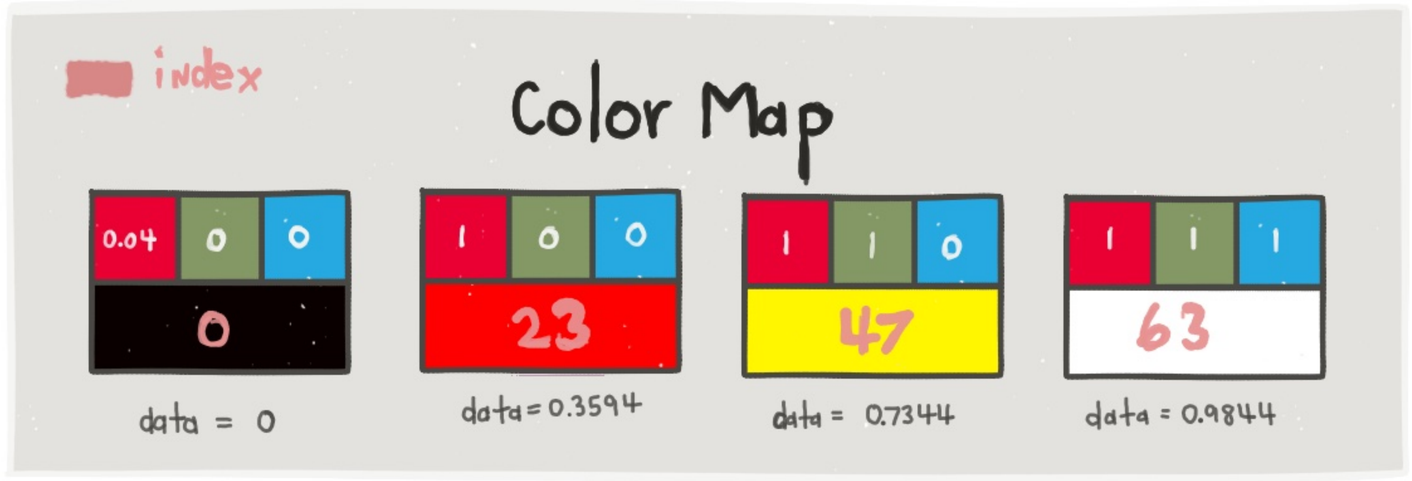
downmix stereo to mono
resample to 22050 Hz with an antialiasing filter of order 8,820,000

audio_to_pitch_via_FB

assume sample rate of 22 050 Hz
assume pitch range of [21:108]
initialize `fs_pitch` array with 128 zeros
initialize `fs_index` array with 128 zeros
fs_pitch[21:59] = 882
fs_pitch[60:95] = 4410
fs_pitch[96:120] = 22050
fs_index[21:59] = 3
fs_index[60:95] = 2
fs_index[96:120] = 1
create copies of signal at 4410 Hz and 882 Hz
calculate the total number of frames and their start/stop locations

seg_pcm_start and seg_pcm_stop hold max Fs sample indices, one per frame
seg_pcm_NUM is the total number of frames

iterate through the range of pitch values (p) performing:
1) get signal `x` according to `fs_index`
2) forward-back filter with corresponding a and b coefficients in filter bank
3) square all values in the filtered signal storing as `f_square`
4) Iterate through range `w=1:winLenSTMSP` performing:
   a) calculate `factor` as input sample rate divide by the `fs_pitch` entry corresponding to the current pitch
   b) Iterate through range `k=1:seg_pcm_NUM{w}` performing:
      i) calculate start as $\lceil \frac{seg\_pcm\_start\{w\}(k)}{22050} \cdot fs\_pitch(p) \rceil$
      ii) calculate stop as $\lfloor \frac{seg\_pcm\_stop\{w\}(k)}{22050} \cdot fs\_pitch(p) \rfloor$
      iii) set `f_pitch_energy{w}(p,k)` by calculating the sum of f_square from start to stop and multiplying by `factor`



Data	R	G	B
0.5911	1	0.5804	0
0.137	0.3725	0	0
0.3523	0.9569	0	0
0.175	0.498	0	0
0.2519	0.7059	0	0
0.01684	0.08235	0	0
0.05741	0.1647	0	0
0.6173	1	0.6667	0
0.05508	0.1647	0	0
0.03619	0.1216	0	0
0.07619	0.2078	0	0
0.1372	0.3725	0	0

```
func color(for value: Float64) -> UIColor {
    let index = (value * 64).rounded(.down)
    return colorMap[index]
}
```

Chroma Vector Quantization

Given a chroma vector $V_n \in [0:1]^{12}$, where n is the vector's frame, the quantization function τ maps V_n such that $V_n \in [0:4]^{12}$.

let $c_i = V_n[i]$ for $i \in [0:11]$

$$\tau(c_i) = \begin{cases} 0 & \text{for } c_i < 0.05 \\ 1 & \text{for } c_i < 0.1 \\ 2 & \text{for } c_i < 0.2 \\ 3 & \text{for } c_i < 0.4 \\ 4 & \text{for } c_i \leq 1 \end{cases}$$

pitch_to_CENS.m

$seg\_NUM = 321$

f_chroma_energy = chroma band energies for each frame

$$f\_chroma\_energy\_distr = \sum_{\substack{c \in [1:12],\\ k \in [1:seg\_NUM]}} \begin{cases} \dfrac{f\_chroma\_energy_{ck}}{\sum_{c=1}^{12} f\_chroma\_energy_{ck}} & \text{for } \sum_{c=1}^{12} f\_chroma\_energy_{ck} > NormThresh \\ 0 & \text{for } \sum_{c=1}^{12} f\_chroma\_energy_{ck} < NormThresh \end{cases}$$

$$f\_stat\_help = \sum_{\substack{c \in [1:12],\\ k \in [1:seg\_NUM]}} \left(\sum_{i=1}^{4} \begin{cases} quantWeights_i & \text{for } f\_chroma\_energy\_dist_{ck} > quantSteps_i \\ 0 & \text{for } f\_chroma\_energy\_dist_{ck} < quantSteps_i \end{cases} \right)$$

Variables

$midiMin = 21$
$midiMax = 108$
$winLengthSmooth = 21$
$downsampSmooth = 5$
$inputFeatureRate = 10$
$quantSteps = [0.4,\ 0.2,\ 0.1,\ 0.05]$
$quantWeights = [0.25,\ 0.25,\ 0.25,\ 0.25]$
$normThresh = 0.001$

f_feature — smoothDownsampleFeature.m

$stat\_window$ = normalized Hanning window of length $winLengthSmooth$

$f\_feature\_stat$ = f_feature appling stat_window as a FIR filter with decimation factor $downsampSmooth$

$$stat\_NUM = \left\lceil \frac{seg\_NUM}{downsampSmooth} \right\rceil$$

$$cut = \left\lfloor \frac{winLengthSmooth - 1}{2 \cdot downsampSmooth} \right\rfloor$$

$f\_feature\_stat = f\_feature\_stat[1:12][cut : stat\_NUM + cut]$

$$new\_feature\_rate = \frac{inputFeatureRate}{downsampleSmooth}$$

f_feature — normalizeFeature.m

$$f\_featureNorm = \sum_{c,k} \begin{cases} \dfrac{f\_feature_{ck}}{\| f\_feature_k \|} & \text{for } \| f\_feature_k \| > NormThresh \\ 0.288675134b & \text{for } \| f\_feature_k \| > NormThresh \end{cases} \quad \text{for } c \in [1:12],\ k \in [1:stat\_NUM]$$

f_CENS

Discrete Cosine Transform

Type 2 $$\sum_{k=0}^{\ell}\left(\sum_{j=0}^{\ell} A_j \cos\left(\frac{k(j+0.5)\pi}{\ell}\right)\right)$$

Type 3 $$\sum_{k=0}^{\ell}\left(\frac{A_0}{2}+\sum_{j=1}^{\ell} A_j \cos\left(\frac{(k+0.5)j\pi}{\ell}\right)\right)$$

Type 4 $$\sum_{k=0}^{\ell}\left(\sum_{j=0}^{\ell} A_j \cos\left(\frac{k(j+0.5)\pi}{\ell}\right)\right)$$

Elliptical Order

complete elliptic integral of the first kind K:

$$K(k) = \int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{1-k^2\sin^2\theta}} = \int_0^1 \frac{dt}{\sqrt{(1-t^2)(1-k^2t^2)}}$$

as a power series:

$$K(k) = \frac{\pi}{2}\sum_{n=0}^{\infty}\left(\frac{(2n)!}{2^{2n}(n!)^2}\right)^2 k^{2n} = \frac{\pi}{2}\sum_{n=0}^{\infty}\left(P_{2n}(0)\right)^2 k^{2n}$$

where P_n is the Legendre polynomials, which is equivalent to:

$$K(k) = \frac{\pi}{2}\left(1 + \left(\frac{1}{2}\right)^2 k^2 + \left(\frac{1\cdot 3}{2\cdot 4}\right)^2 k^4 + \ldots + \left(\frac{(2n-1)!!}{(2n)!!}\right)^2 k^{2n} + \ldots\right)$$

computed efficiently in terms of the arithmetic-geometric mean:

$$K(k) = \frac{\frac{\pi}{2}}{agm\left(1, \sqrt{1-k^2}\right)}$$

arithmetic-geometric mean of two positive real numbers x and y:

compute the arithmetic (a_1) and geometric (g_1) means of x and y,

$$a_1 = \frac{1}{2}(x+y)$$

$$g_1 = \sqrt{xy}$$

then compute with a_1 substituted for x and g_1 substituted for y. By repeating this process, two sequences a_n and g_n are defined:

$$a_{n+1} = \frac{1}{2}(a_n + g_n)$$

$$g_{n+1} = \sqrt{a_n g_n}$$

these two sequences converge to the same number, the arithmetic-geometric mean or $agm(x,y)$

calculating **eliptic filter order** for given values $\varepsilon, A, \Omega_p, \Omega_s$:

$$n = \frac{K(k)K\left(\sqrt{1-k_1^2}\right)}{K(k_1)K\left(\sqrt{1-k^2}\right)}$$

where $K(\bullet)$ is the complete elliptic integral of the first kind

From Theory and Application of Digital Signal Processing by Rabiner and Gold

passband and stopband edge frequencies: Ω_p, Ω_s

passband and stopband gains: G_p, G_s $\quad G = \frac{1}{\sqrt{1+\varepsilon^2}} = 10^{-\frac{A}{20}}$

ripple parameters: $\varepsilon_p, \varepsilon_s$ $\quad \varepsilon = \sqrt{G^{-2}-1} = \sqrt{10^{\frac{A}{10}}-1}$

attenuation (dB): A_p, A_s $\quad A = -20\log_{10} G = 10\log_{10}(1+\varepsilon^2)$

selectivity parameter: k $\quad k = \frac{\Omega_p}{\Omega_s}$

discrimination parameter: k_1 $\quad k_1 = \frac{\varepsilon_p}{\varepsilon_s}$

complete elliptic integral of the first kind: $K = \int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{1-k^2\sin^2\theta}}$

degree equation: $F_N(k^{-1}) = k_1^{-1}$ or $N\frac{K'}{K} = \frac{K_1'}{K_1}$

$F_N(w) = cd(NuK_1, k_1)$, $w = cd(uK, k)$

$cd(x,k)$ denotes the Jacobian elliptic function with modulus k and real quarter-period K

magnitude response $\quad |H(\Omega)|^2 = \frac{1}{1+\varepsilon_p^2 F_N^2\left(\frac{\Omega}{\Omega_p}\right)} = \frac{1}{1+\frac{\varepsilon_s^2}{F_N^2\left(\frac{\Omega_s}{\Omega}\right)}}$

$k' = \sqrt{1-k^2}$

$K' = K$ with $k = k'$ $\qquad \Omega = \frac{2\pi f}{f_s}$

Landen Transformations

$$k_n = \left(\frac{k_{n-1}}{1+k'_{n-1}}\right)^2, \quad n = 1, 2, \ldots, M$$

Chord Recognition

Template-Based

Map the 12 chromas to a bit field

Measuring similarity using the inner product of normalized vectors:

$$s(x,y) = \frac{\langle x|y\rangle}{\|x\| \cdot \|y\|}$$

Evaluating Results

TP = True Positive, Correctly recognized chord
FP = False Positive, Incorrectly recognized chord
FN = False Negative, Failure to recognize a chord
P = Precision R = Recall F = F-Measure

$$P = \frac{\#TP}{\#TP + \#FP} \qquad R = \frac{\#TP}{\#TP + \#FN} \qquad F = \frac{2 \cdot P \cdot R}{P + R}$$

Incorporating Harmonics

Along with each note, consider the first 8 partials.

For example the first 8 partials for C would be C, C, G, C, E, G, Bb, C.

Define constant α and assume energy of the k^{th} partial is α^{k-1}.

A chroma vector for just a C note would then be the following:

$$t_C^h = (1 + \alpha + \alpha^3 + \alpha^7,\ 0,\ 0,\ 0,\ \alpha^4,\ 0,\ 0,\ \alpha^2 + \alpha^5,\ 0,\ 0,\ \alpha^6,\ 0)$$

C Db D Eb E F Gb G Ab A Bb B

Add the vectors for each note to get the template for the composed chord.

$$t_{CMaj}^h = t_C^h + t_E^H + t_G^H$$

let $\alpha = 0.9$

$t_C^H = (3.1072969, 0, 0, 0, 0.6561, 0, 0, 1.40049, 0, 0, 0.531441, 0)$

partials for E would be E E B E Ab B D E
and for G: G G D G B D F G

Bit Representation of Chord Patterns

12 intervals (interval 1 is implied)
Each interval has 4 possible states requiring 2 bits
possible states:
- 0 0 interval is not included
- 1 0 interval is included
- 0 1 flattened interval is included
- 1 1 sharpened interval is included

uses the 24 least significant bits in a UInt32 value

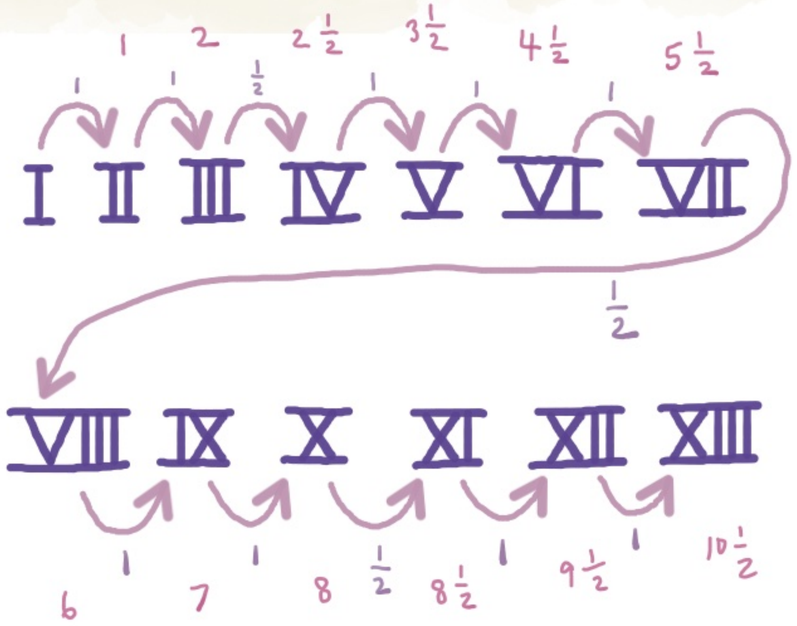
example 1: 7sus4 = (4, 5, b7) = 00000000000000000000100101000000

convert to chroma for a root of **C**: [C, F, G, Bb]

C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
I		II		III	IV		V		VI		VII
C					F		G			Bb	

and for a root of **Db**: [Db, Gb, Ab, B]

C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
VII	I		II		III	IV		V		VI	
	Db					Gb		Ab			B



example 2: Maj13#11 = (3, 5, 7, 9, #11, 13) = 00000000100011001000100010001000

convert to chroma for a root of **C**: [C, E, G, B, D, Gb, A]

C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
I		II		III	IV		V		VI		VII
VIII		IX		X	XI		XII		XIII		
C		D		E		Gb		G		A	B

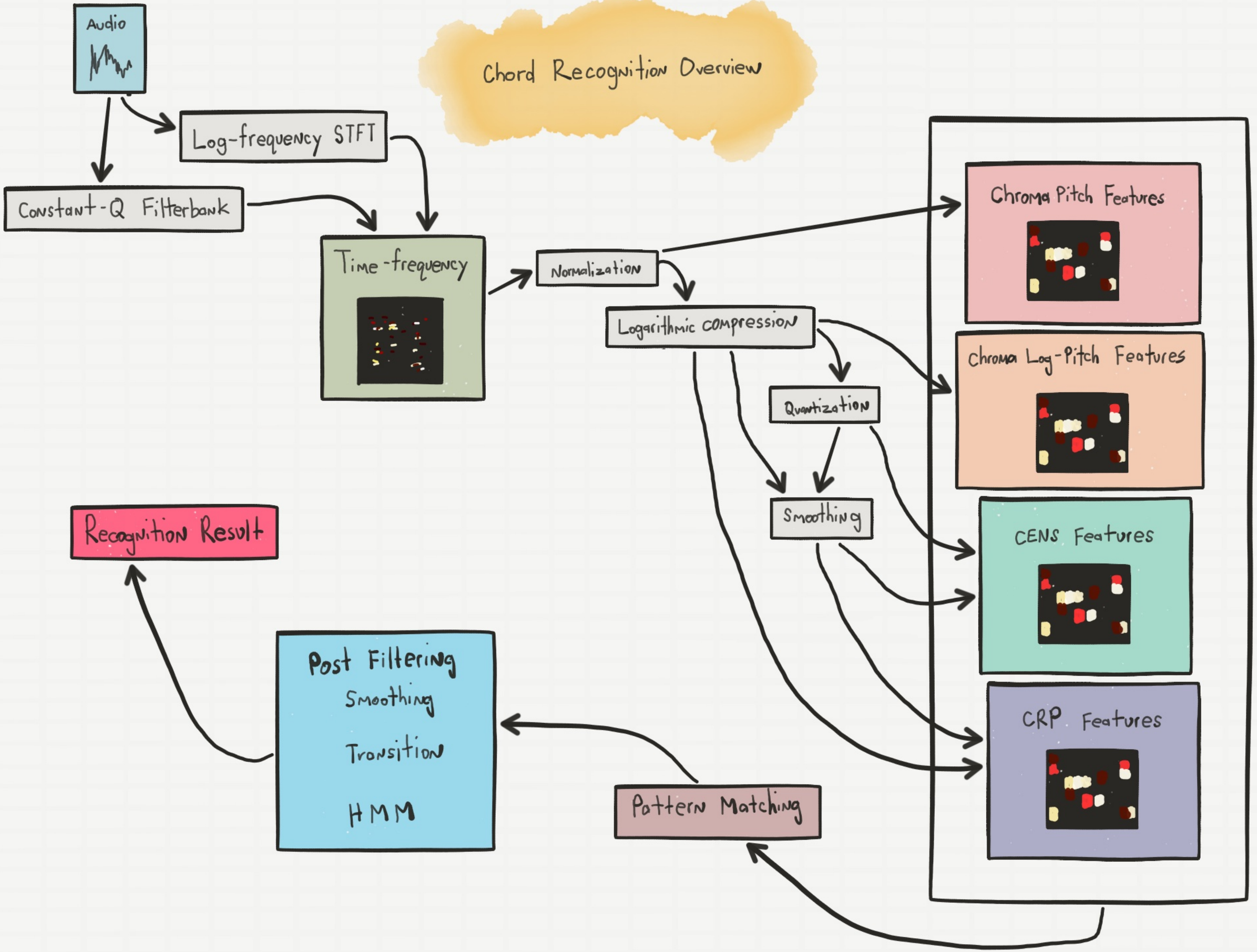
and for a root of **Db**: [Db, F, Ab, C, Eb, G, Bb]

C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B
VII	I		II		III	IV		V		VI	
	VIII		IX		X	XI		XII		XIII	
	Db				F			Ab			
C			Eb				G			Bb	

Chroma Template Vectors

C vector = (	3.107297	0	0	0	0.6560999	0	0	1.40049	0	0	0.5314409	0	)
E vector = (	0	0	0.5314409	0	3.107297	0	0	0	0.6560999	0	0	1.40049	)
G vector = (	0	0	1.40049	0	0	0.5314409	0	3.107297	0	0	0	0.6560999	)
CEG vector = (	3.107297	0	1.9319309	0	3.7633969	0.5314409	0	4.507787	0.6560999	0	0.5314409	2.0565899	)

Chord Recognition Overview



ChromaToolbox Demo Parameters

Pitch Features
- window length = 4410
- hop size = 2205
- MIDI Range = [21:108]

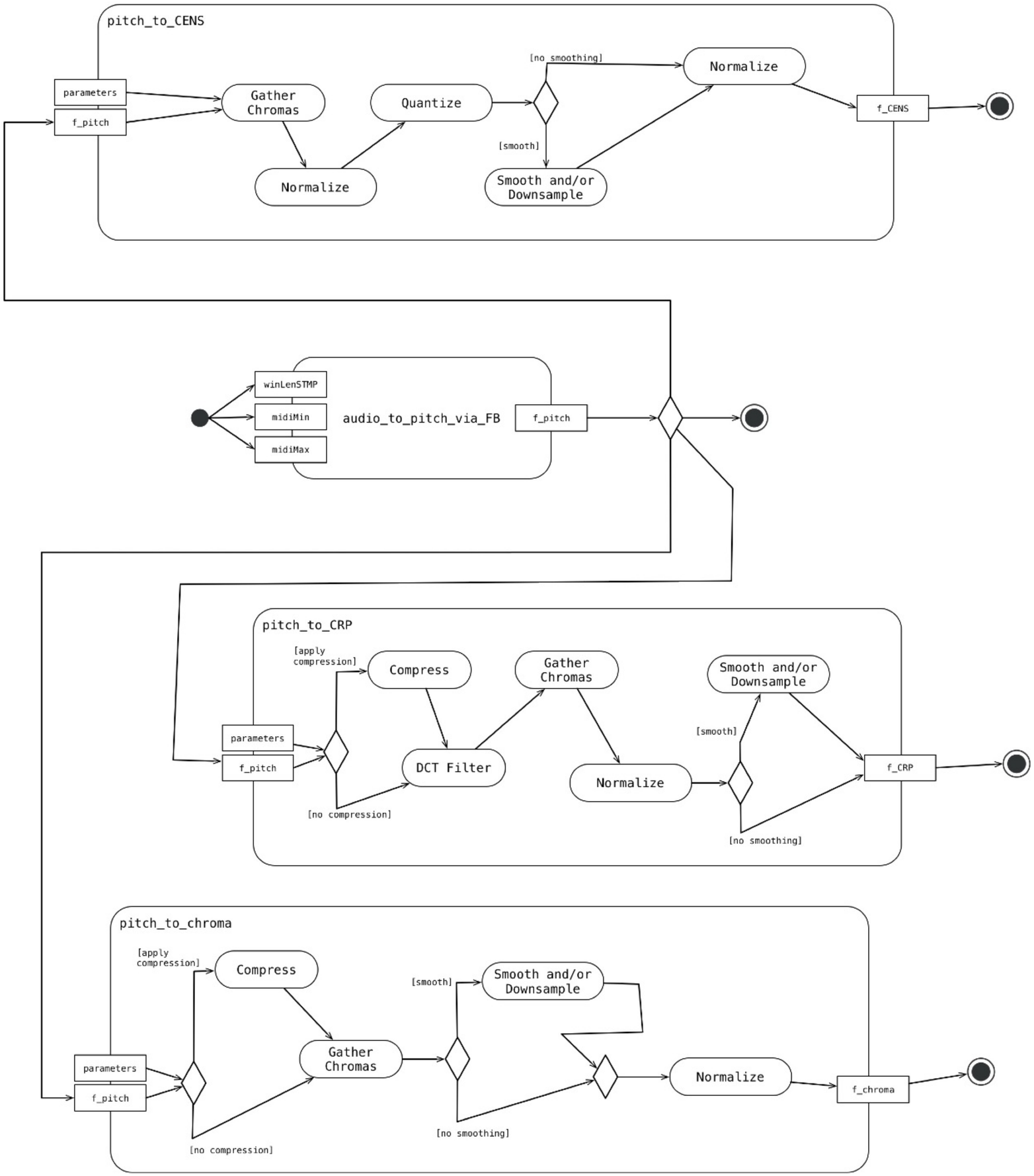
Chroma Pitch
- compress = false
- normalize = true
- norm p = 2
- norm threshold = 0.001
- MIDI Range = [1:120]

Chroma Log-Pitch
- compress = true
- compression factor = 100
- normalize = true
- norm p = 2
- norm threshold = 0.001
- MIDI Range = [1:120]

CENS
- quantization steps = [0.4, 0.2, 0.1, 0.05]
- quantization weights = [0.25, 0.25, 0.25, 0.25]
- norm threshold = 0.001
- smoothing window length = 21
- smoothing downsample = 5
- MIDI Range = [21:108]

CRP1
- kept coefficients = [55:120]
- compress = true
- compression factor = 1000
- norm p = 2
- norm threshold = 0.000001
- smoothing window length = 1
- smoothing downsample = 1

CRP2
- kept coefficients = [55:120]
- compress = true
- compression factor = 1000
- norm p = 2
- norm threshold = 0.000001
- smoothing window length = 21
- smoothing downsample = 5


pitch_to_CENS
parameters
f_pitch
Gather Chromas
Quantize
[no smoothing]
Normalize
f_CENS
[smooth]
Normalize
Smooth and/or Downsample
winLenSTMP
midiMin
midiMax
audio_to_pitch_via_FB
f_pitch
pitch_to_CRP
[apply compression]
Compress
Gather Chromas
Smooth and/or Downsample
parameters
f_pitch
DCT Filter
Normalize
[smooth]
f_CRP
[no compression]
[no smoothing]
pitch_to_chroma
[apply compression]
Compress
[smooth]
Smooth and/or Downsample
parameters
f_pitch
Gather Chromas
Normalize
f_chroma
[no compression]
[no smoothing]

Similarity Scores

Calculate the inner product

otherVector

0.014	0.000	0.027	**0.376**	0.013	0.011	0.000	**0.200**	0.099	0.013	0.000	**0.248**

vector

0.000	0.000	0.120	**0.213**	0.000	0.082	0.000	**0.251**	0.000	0.000	**0.295**	0.038

products

0.000	0.000	0.00324	0.080088	0.000	0.000902	0.000	0.0502	0.000	0.000	0.000	0.009424

$$\sum_{i=0}^{11} \text{products}_i = 0.143854$$

$$+ 0.2 \times 0.85$$

$$+ 0.2 \times 3$$

$$+ 0.2 \text{ high root energy bonus}$$

$$- 0.1$$

$$0.813854$$

Similarity Scores

Chroma with energy > 0.35

Vectors for chords played without the 5th

Chords with identical templates

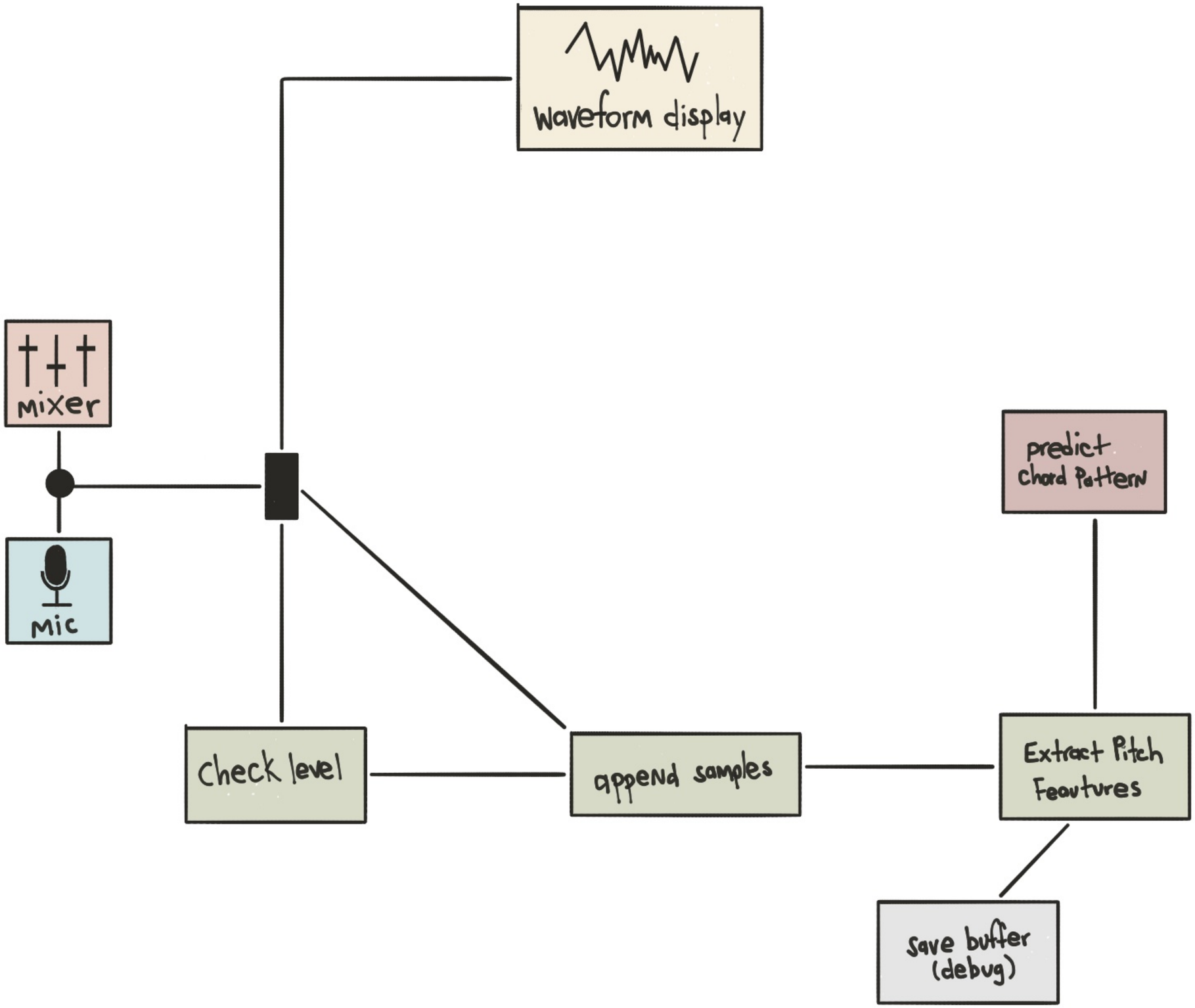
for each chord, which intervals are optional?

Vectors with 85% of energy in two chromas

document guess of how many notes are present

Missing energy for templates chord root

templates where there is energy for all chord chromas.


waveform display
mixer
Mic
predict
Chord Pattern
Check level
append samples
Extract Pitch
Features
Save buffer
(debug)

Signal Processing for WaterfallView

SignalProcessing.FFT

N = 4800

$\log_2 N = 12$

K = 2400

bin count = 2401

perform vDSP_zrip

divide all bins by 2

Move the Nyquist value

take the square roots of the squared magnitudes

doFFT_onAudioBuffer

N = 2048

$\log_2 N = 11$

Multiply with a blackman window of length N (only first $\frac{N}{2}$?)

perform vDsp_fft_zip

calculate squared magnitudes

divide the squared magnitudes by 2

SignalProcessing.STFT

N = `windowSize` (default = 8820)

multiply by a normalized Hann window of size N

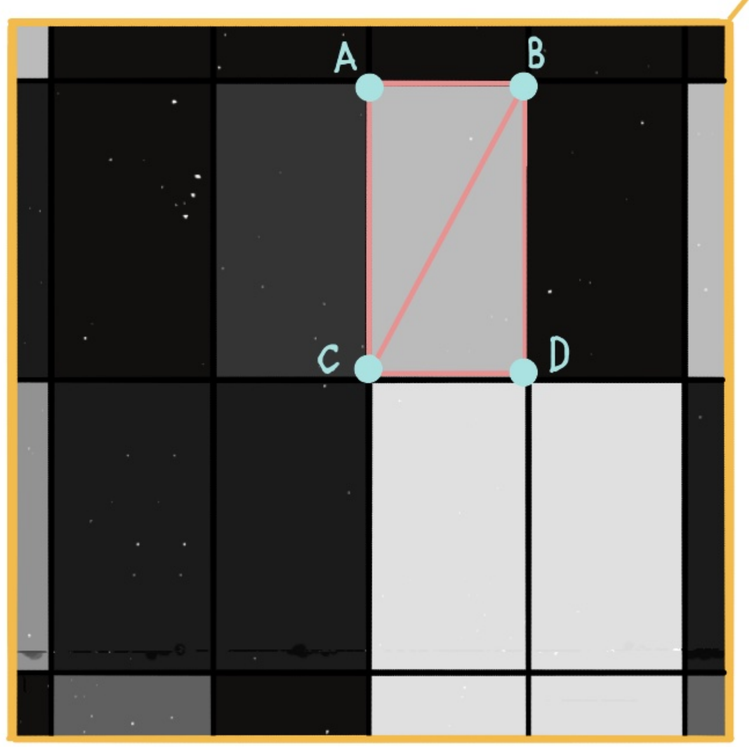
Variables

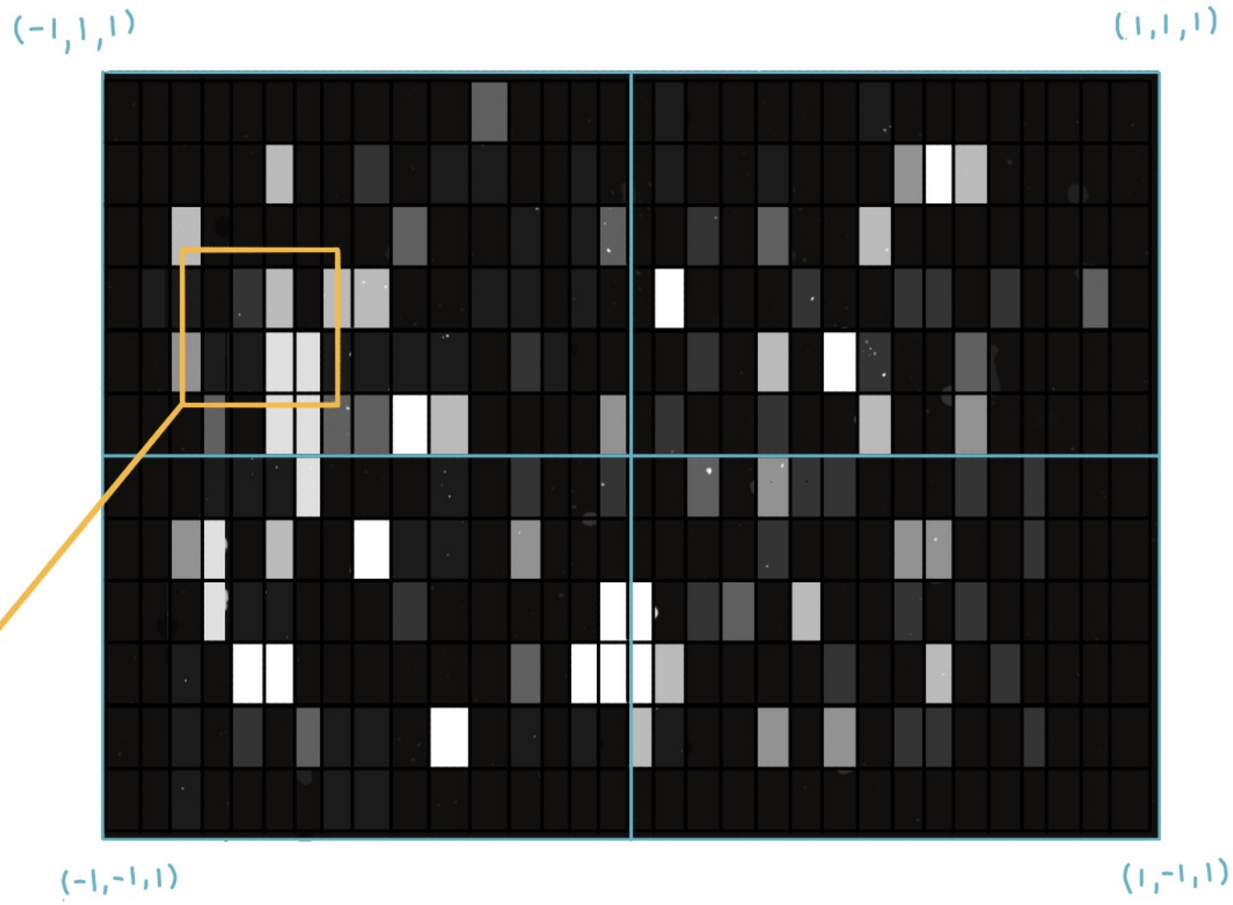
spectrumCount the total # rows = 12

binCount the total # columns = 32

$$h = \frac{2}{\text{spectrumCount}}$$

$$w = \frac{2}{\text{binCount}}$$





b = bin index s = spectrum index s' = spectrum count - s

$A = (bw-1, sh-1)$

$B = (A_x + w, A_y)$

$C = (A_x, A_y - h)$

$D = (B_x, C_y)$

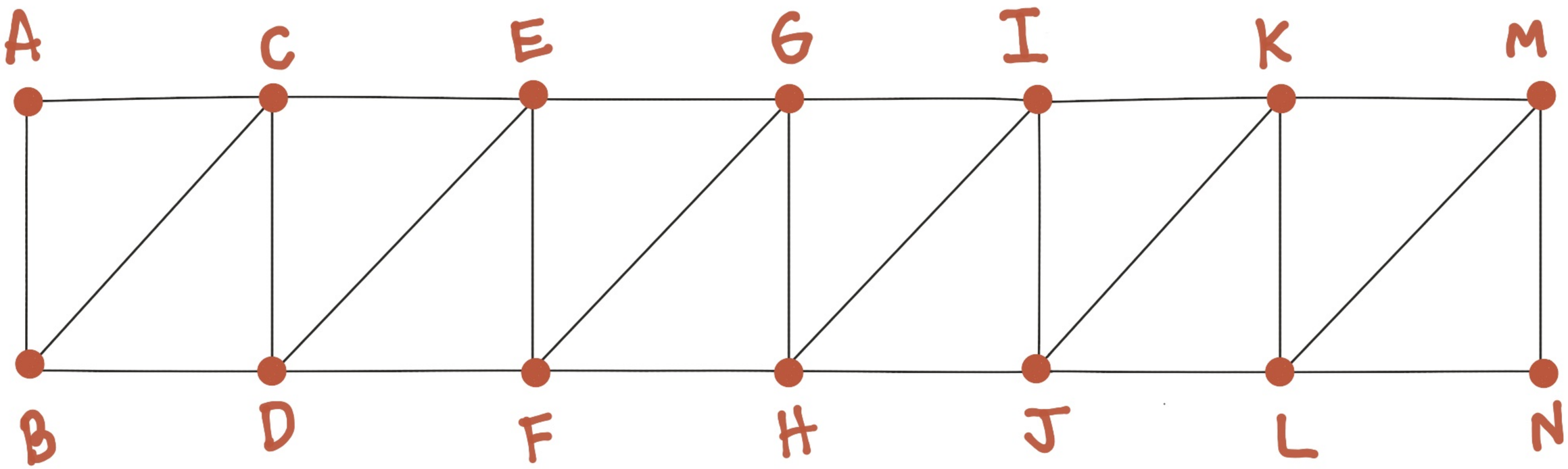
b=5 s=3 s'=9 $w = \frac{1}{16}$ $h = \frac{1}{6}$

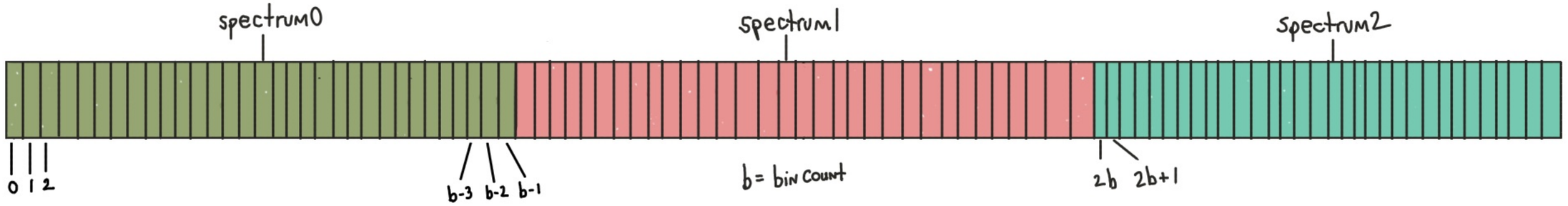
$A = (-\frac{11}{16}, \frac{1}{2})$

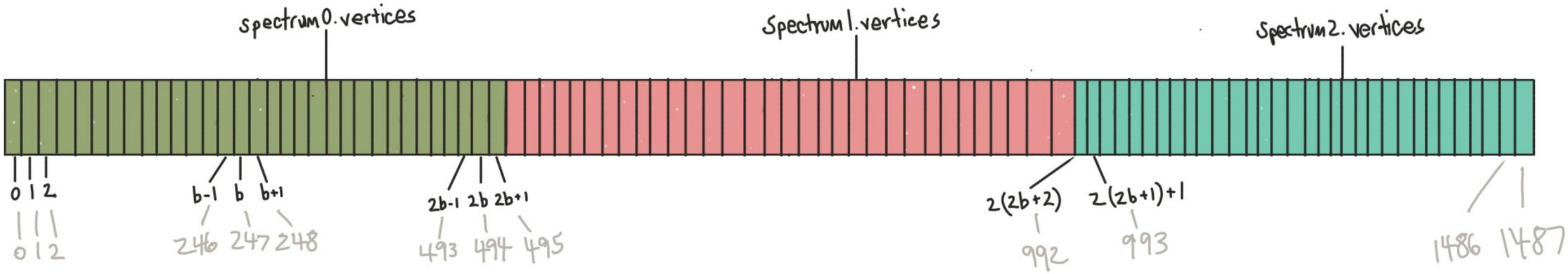
$B = (-\frac{10}{16}, \frac{1}{2})$

$C = (-\frac{11}{16}, \frac{1}{3})$

$D = (-\frac{10}{16}, \frac{1}{3})$


A
C
E
G
I
K
M
B
D
F
H
J
L
N





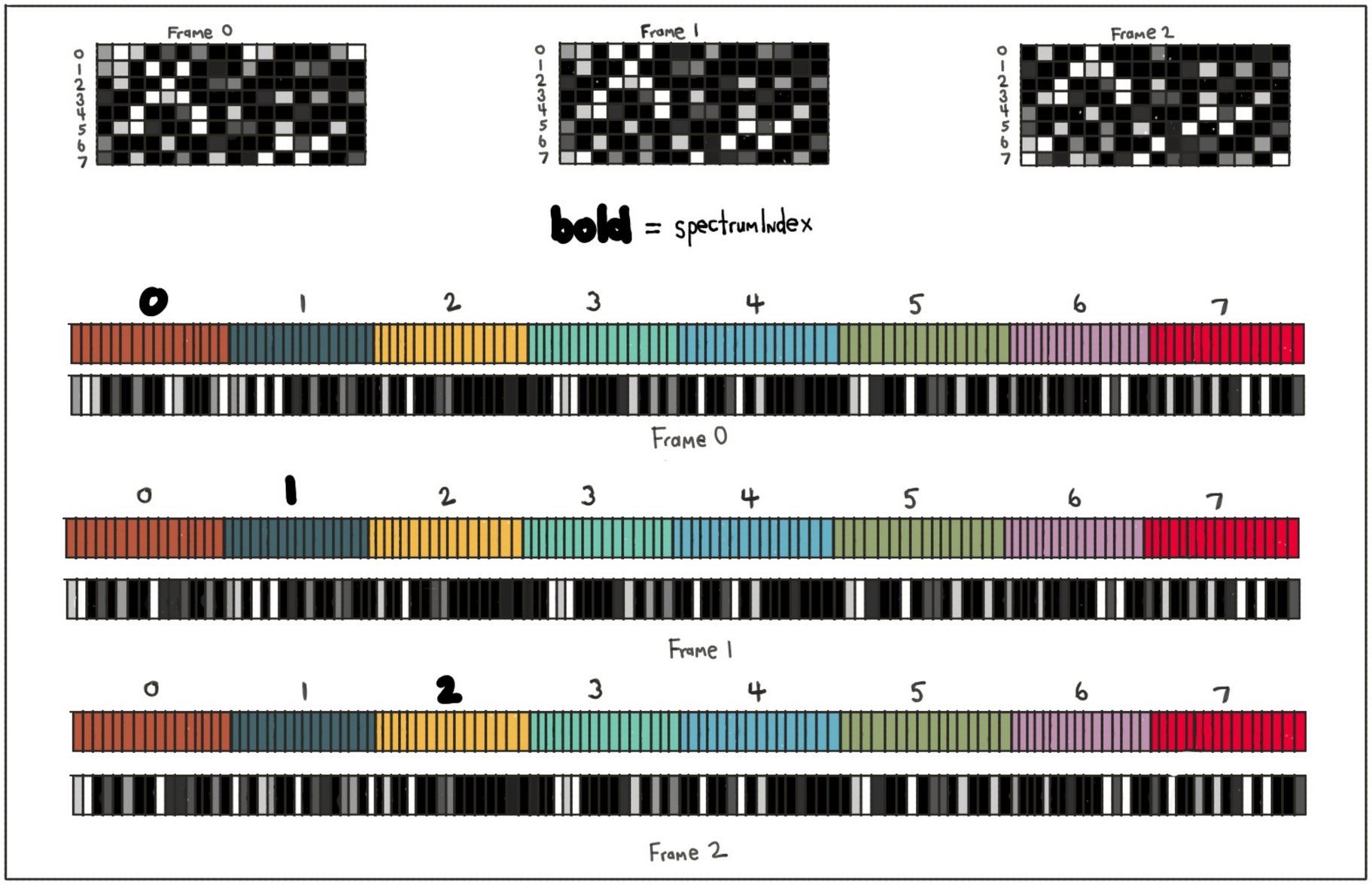
V = vertices per spectrum $= 2b + 2$

$$\text{bin}(\text{vid}) = \text{MIN}\left(\frac{\text{vid Mod } V}{2}, b-1\right)$$

$$\text{spectrum}(\text{vid}) = \frac{\text{vid}}{V}$$

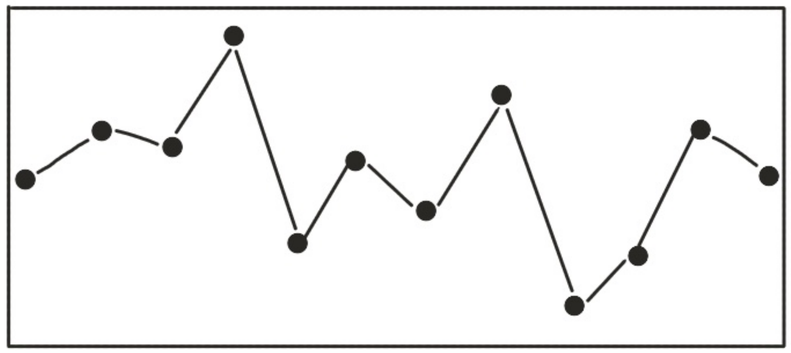
let $b = 247$, $V = 496$

vid	bin
326	163
492	246
494	246
496	0



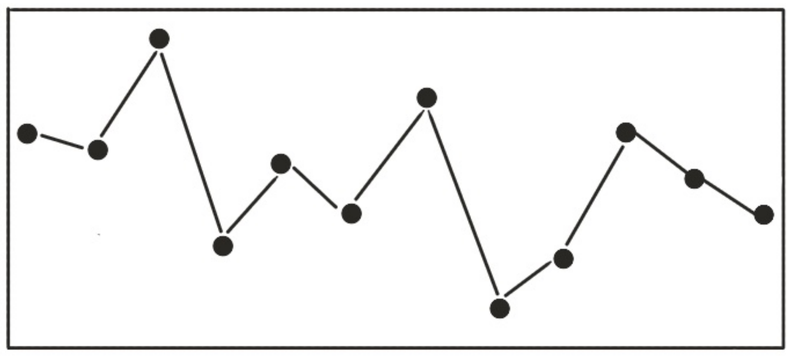

```
float4 vertex(vid) {
  sid = vid / spectrumCount
  binOffset = (vid % binCount) / 2
  bid = min(binOffset, binCount - 1)
  isTop = vid % 2 == 0
  w = 2 / binCount
  h = 2 / spectrumCount
  Δ = sid - spectrumIndex
  row = (spectrumCount + Δ) % spectrumCount
  y = 1 - h * row - (isTop ? 0 : h)
  x = -1 + w * binOffset
  return float4(x, y, 0, 1)
}
```

t = top = index of the oldest sample



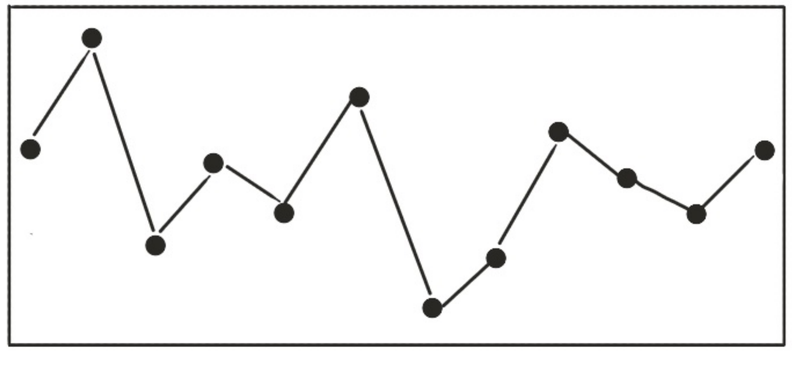
t

0.0	0.3	0.2	0.9	-0.4	0.1	-0.2	0.4	-0.8	-0.5	0.3	0.0



t

-0.2	0.3	0.2	0.9	-0.4	0.1	-0.2	0.4	-0.8	-0.5	0.3	0.0



t

-0.2	0.1	0.2	0.9	-0.4	0.1	-0.2	0.4	-0.8	-0.5	0.3	0.0

```
vertex VertexOut func waveformVertex( uint vid [[ vertex_id ]],
                                      WaveformVertexArguments &arguments [[ buffer(0) ]])
{
    uint top = arguments.top;
    uint sampleCount = arguments.sampleCount;
    device float *samples = arguments.samples;

    VertexOut outVertex;
    float y = samples[vid];
    float xOffset = 2.0f / float(sampleCount);
    uint index = vid > top ? vid - top : (vid + sampleCount) - top;

    float x = -1.0f + float(index) * xOffset;

    outVertex.position = float4(x, y, 0.0f, 1.0f);

    outVertex.color = float4(1.0f, 1.0f, 0.0f, 1.0f);

    return outVertex;
}
```

Incoming Audio Samples

0.03	-0.01	0.2	0.1

Acquire enough new samples to fill the FFT window and calculate the pitch spectrum

Pitch Spectrums

0.05	0.1	0.002	0.8
0.001	0.4	0.16	0.6
0.9	0.0012	0.31	0.02
0.06	0.003	0.23	0.004

Spectrum Buffer

0.05	0.01	0.002	0.8
0.001	0.4	0.16	0.6
0.036	0.0006	0.332	0.91
0.06	0.003	0.23	0.004
0.05	0.01	0.002	0.8
0.001	0.4	0.16	0.6
0.9	0.0012	0.31	0.02
0.06	0.003	0.23	0.004
0.05	0.01	0.002	0.8
0.44	0.09	0.061	0.2
0.036	0.0006	0.332	0.91
0.06	0.003	0.23	0.004

Copy the New collection of spectrums into the Next frame's buffer

$$displayCount = samplesConsumed \times framesPerSample = 4410$$

$$samplesConsumed = \frac{sampleRate}{framesPerSecond} = 735$$

sampleRate = 44,100

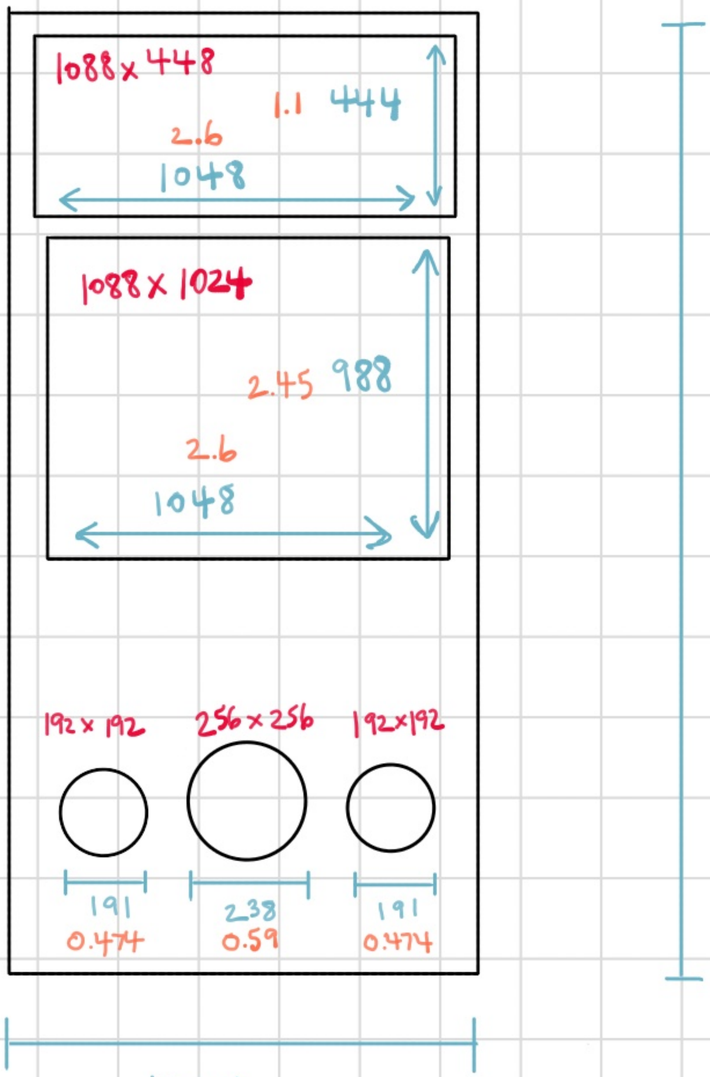
framesPerSecond = 60

framesPerSample = 6

Frame	Samples
0	0 ..< 4410
1	735 ..< 5145
2	1470 ..< 5880
3	2205 ..< 6615
4	2940 ..< 7350
5	3675 ..< 8085

px IN Tx

1152 x 2456

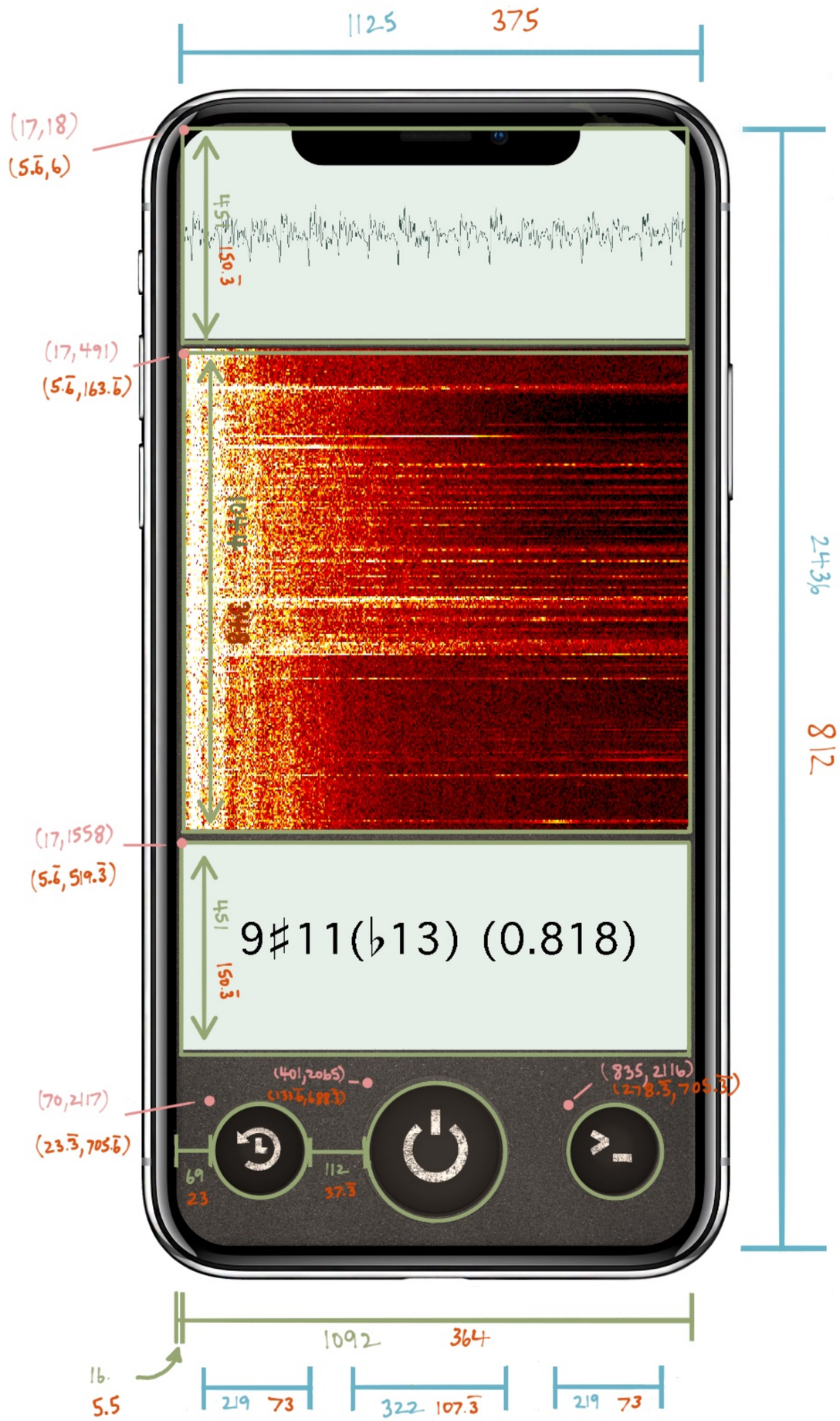


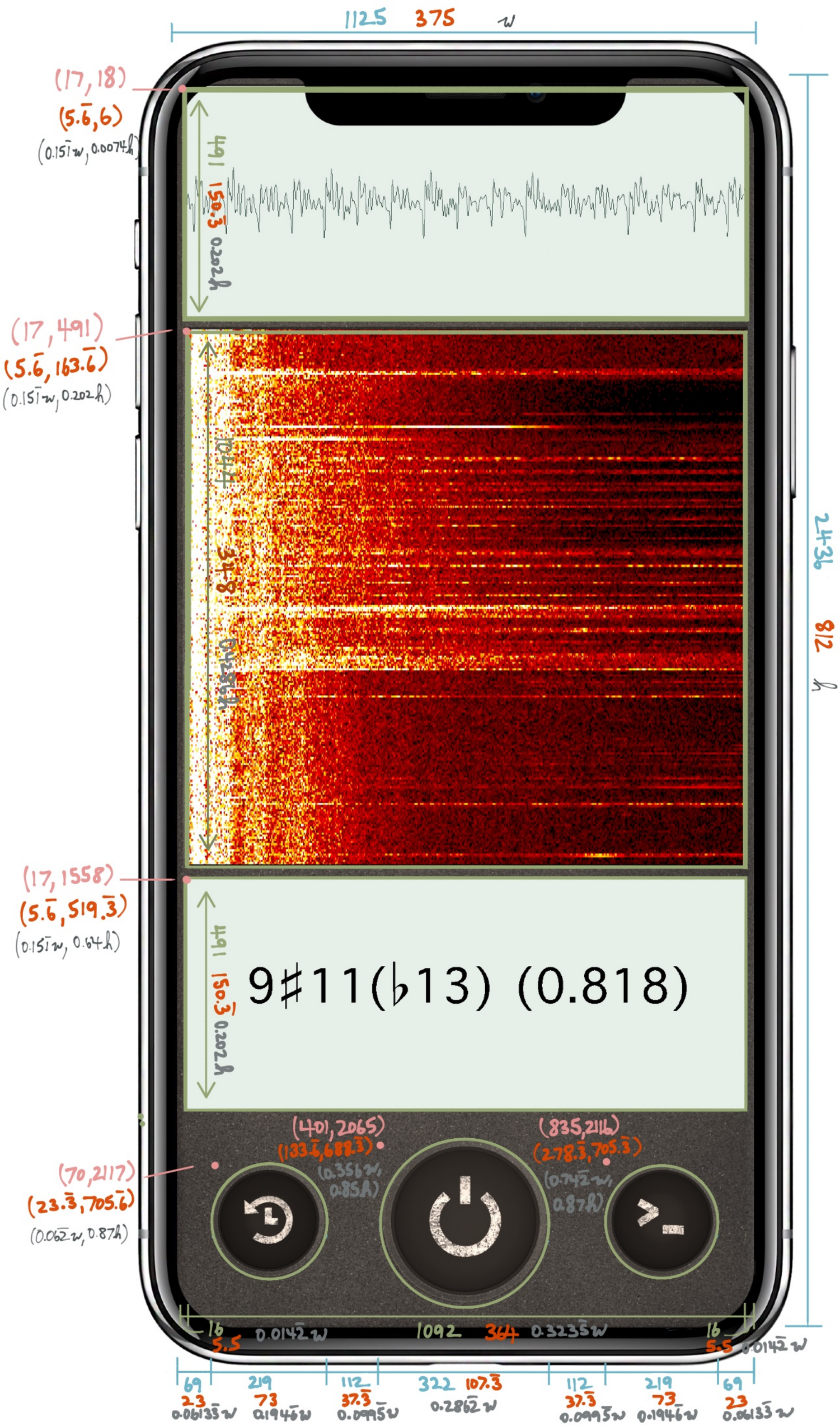
1125
2.79

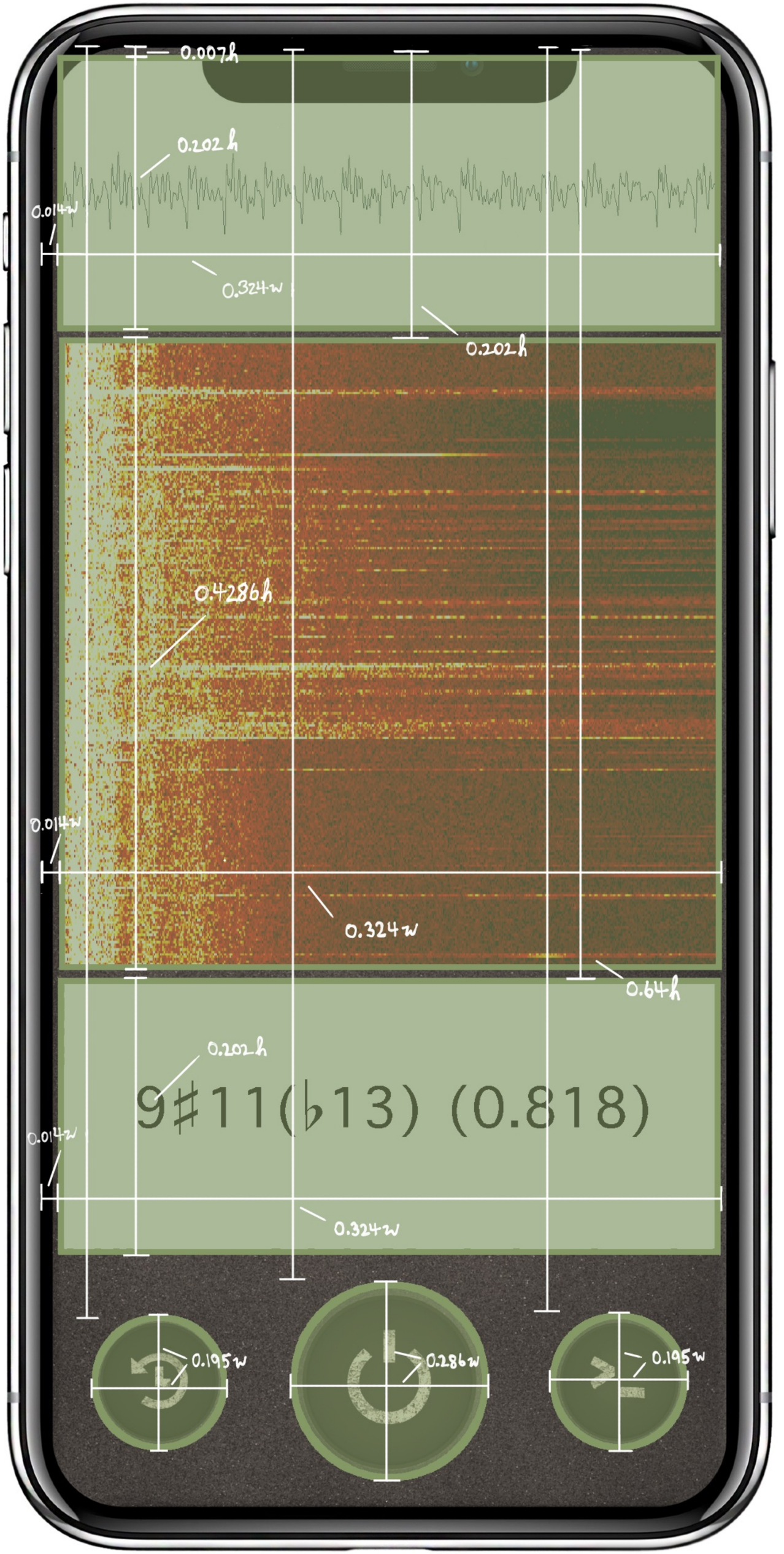
2436 6.044

$$\frac{2436}{6.044} = 403.04434414957$$

$$\frac{1125}{2.79} = 403.2258064516$$

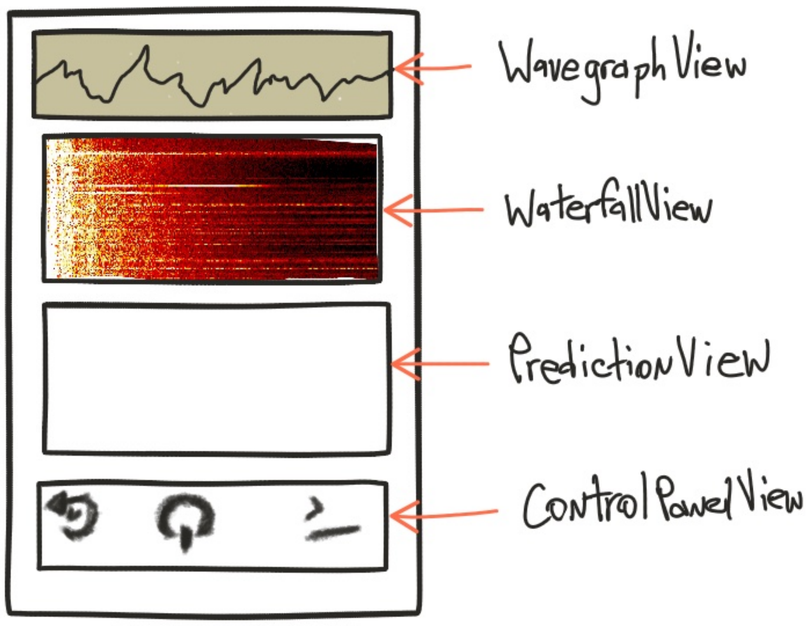

1125
375
(17,18)
(5.6̄,6)
451
150.3̄
(17,491)
(5.6̄,163.6̄)
1558
1067
(17,1558)
(5.6̄,519.3̄)
451
150.3̄
9♯11(♭13) (0.818)
(401,2065)
(133.6̄,688.3̄)
(835,2116)
(278.3̄,705.3̄)
(70,2117)
(23.3̄,705.6̄)
69
23
112
37.3̄
2436
812
1092
364
16
5.5
219
73
322
107.3̄
219
73


1125 375 w
(17, 18)
(5.6̄, 6)
(0.15̄w, 0.007h)
491 159.3̄ 0.202h
(17, 491)
(5.6̄, 163.6̄)
(0.15̄w, 0.202h)
1044
348
(17, 1558)
(5.6̄, 519.3̄)
(0.15̄w, 0.64h)
491 159.3̄ 0.202h
9♯11(♭13) (0.818)
(401, 2065)
(133.3̄, 688.3̄)
(0.356w, 0.85h)
(835, 2116)
(278.3̄, 705.3̄)
(0.74̄2w, 0.87h)
(70, 2117)
(23.3̄, 705.6̄)
(0.06̄2w, 0.87h)
2436
812
16 5.5 0.01̄42w
1092 364 0.3235̄w
16 5.5 0.01̄42w
69 23 0.0613̄3w
219 73 0.1946̄w
112 37.3̄ 0.0995̄w
322 107.3̄ 0.28̄62w
112 37.3̄ 0.0995̄w
219 73 0.1946̄w
69 23 0.0613̄3w


0.007h
0.202 h
0.014w
0.324w
0.202h
0.4286h
0.014w
0.324w
0.64h
0.202h
9♯11(♭13) (0.818)
0.014w
0.324w
0.195w
0.286w
0.195w

Front Burner

- UIButton subclass for custom animation
- Training capability added to MLKit
- Aggregate results from testing overlapping audio clips
- Incorporate known characteristics into prediction logic
- Implement 3D model for custom drawing
- Tuner functionality
- Chord library functionality



Pattern	Intervals	Confidence
m (add 9)		0.9352

Adding Audiobus Support

- Make sure audio session is configured to "mix with others" before becoming active

- Add Audiobus SDK
- Enable background and inter-app audio

- Life Cycle Management:
  - ensure running and active audio session
    - stop audio system when disconnected from Audiobus
    - never stop audio system while Audiobus controller is connected
    - Implement suggested background policy

- Set up launch URL
- Register and generate API key
- Add Audiobus Controller instantiation
- Create sender and receiver Audiobus ports adding component descriptions to Info.plist
- Update the Audiobus registry

Audio Manager