Stricto Sensu Editora

CRIAÇÃO MEDIADA TECNOLOGICAMENTE: O *FAST FOURIER TRANSFORM*

Charles K. Neimog[1] e Rodolfo Coelho de Souza[1]

1. Universidade de São Paulo, São Paulo, Brasil.

RESUMO
Este texto tem o objetivo de evidenciar propriedades do Fast Fourier Transform (FFT) que podem ter impacto na estética e condução do processo criativo mediado pelo computador. Ao final do texto apresentamos uma biblioteca de OpenMusic com documentação em português (ainda em desenvolvimento) que permite visualizar e entender partes do processo do FFT, contribuindo didaticamente uma vez que tem seu código comentado e aberto, diferente de bibliotecas compiladas como a OM-SuperVP e OM-pm2.
Palavras-chave: Fast Fourier Transform, OpenMusic e Música Eletroacústica.

ABSTRACT
This article highlights some properties of the Fast Fourier Transform (FFT) that may impact the aesthetic and conduction of the creative process mediated by the computer. At the end of the article, we will present an OpenMusic library (a work in progress) that allows the visualization and understanding of each separate part of the FFT process, contributing didactyly with your open and commented code, differently from OM-SuperVP and OM-pm2 libraries (distributed in binaries).
Keywords: Fast Fourier Transform, OpenMusic and Electroacoustic Music.

1. INTRODUÇÃO

A transformada de Fourier foi concebida pelo matemático Joseph Fourier (1768-1830) a partir do axioma de que: “qualquer vibração periódica, por mais complicada que pareça, pode ser construída a partir de senoides cujas frequências são múltiplas inteiras de uma frequência fundamental, escolhendo as amplitudes e fases adequadas” (LOY, 2007). O *Fast Fourier Transform* (FFT), um algoritmo de otimização da Transformada de Fourier, no áudio,

é aplicado na transposição de alturas, na compressão de dinâmicas, no *time-stretching*, em *reverbs* por convolução, em determinados tipos de espacialização, entre outras.

A despeito de suas reconhecidas potencialidades, o FFT também tem limitações principalmente pois, na maioria das vezes, o áudio não se 'encaixa' no axioma de uma vibração totalmente periódica. Entendendo que, mesmo havendo alterações nos currículos universitários, a tecnicidade dos processos algoritmos ainda não faz parte do *metier* do(a) compositor(a), acreditamos que a padronização de parâmetros de análise, o uso de interfaces gráficas e os complexos processos matemáticos envolvidos, podem contribuir para uma certa formatação das características sônicas da música eletroacústica. Por este motivo, conhecer os detalhes de parâmetros do FFT, saber manipulá-los, e de certa forma, aprender a ouvir as suas características pode contribuir para evitar (ou uso consciente) características como os descritos por Campos Júnior (2005) no caso do uso do aplicativo *AudioSculpt* na ressíntese de amostras sonoras ruidosas.

Nossa pesquisa almeja dar uma contribuição didática ao problema, desenvolvendo uma biblioteca de *OpenMusic*, comentada em português, que permite visualizar e entender partes do processo do FFT, tais como a conversão de um áudio para números e os cálculos para obter as frequências (Hz) e as amplitudes. A biblioteca apresentada está em linguagem *Lisp*, que diferente de bibliotecas como a OM-SuperVP e OM-pm2 em C, permite ver o código e entender o passo dos processos (no OpenMusic pressionando a tecla *e* com o objeto selecionado). Isso não acontece nas bibliotecas supracitadas pois ambas são proprietárias (Ircam) assim como são compiladas (distribuídas em linguagem de máquina), o OpenMusic, neste caso, somente usa dados processados pelo AudioSculpt (no caso da OM-SuperVP), os processos, o código, os cálculos e os parâmetros internos continuam inacessíveis ao(à) usuário(a).

2. UMA CARACTERIZAÇÃO DO FAST FOURIER TRANSFORM

Apresentada a nossa problemática, no processamento de áudio utilizamos uma implementação específica do FFT, chamada de *Short-Time Fourier Transform* (STFT) (KLINGBEIL, 2009, p. 18). Neste algoritmo, divide-se uma onda sonora em partes, e aplica-se o FFT a cada uma dessas partes. Avançando-se um determinado número (*hop size*) de amostras digitais (*samples*) pode-se obter uma fotografia de um trecho do áudio. No STFT

há dois conceitos primordiais: o tamanho da janela FFT (*FFT size*), que é o total de amostras do sinal sonoro que serão analisados por vez – necessariamente um expoente de 2 –, e a *hop size*, que define a quantidade de amostras da onda sonora que serão avançadas a cada cálculo de FFT, ou seja, de quantas em quantas amostras digitais teremos uma foto espectral do áudio.

Para escolher o tamanho da janela FFT devemos equilibrar duas tendências opostas. Quanto maior a janela, maior a definição da frequência, mas menor a definição da variação dos parciais com o tempo. E vice-versa. Por exemplo, em uma janela FFT de 4096 com uma taxa de amostragem de 44100 Hz teremos uma 'foto' espectral que leva em conta 92.8 milissegundos de áudio (cálculo obtivo por regra de 3).

$$\begin{matrix} 1 & 44100 \\ x & 4096 \end{matrix}$$

Eq. 1

As informações sobre alterações de parciais neste período serão perdidas. Assim podemos afirmar que, de modo geral, em um som com muitas alterações em seus parciais (sons ruidosos ou pouco constantes), o ideal é utilizar janelas FFT menores. Por outro lado, quando temos um som contínuo e estável, podemos utilizar janelas FFT maiores.

Para exemplificar como a escolha da Janela FFT é sugestiva na descrição de um som, sugerimos a audição dos três áudios disponíveis no seguinte link: https://bit.ly/3x68UjE. Note-se que no caso do som de pássaro, a análise padrão do programa *Spear* (com resolução da frequência igual a 40Hz) descreve mal o som proposto pois utiliza uma Janela FFT de 16384 amostras. Produzindo uma "foto espectral" que pareceria suficiente, a cada 550 samples (12 ms.), o FFT leva em conta 372 ms. de áudio não somente 12 ms. como se pode supor. É por este motivo, que o som do pássaro, que contém muitas modificações espectrais, perde muitas alterações que acabam ficando indetectáveis ou por muitas vezes essas alterações são descritas como duas parciais diferentes (por isso o som inarmônico onde se pode ouvir *clusters* em alguns parciais). Por outro lado, com uma janela FFT menor (1024 com a resolução de frequência igual a 345Hz), a descrição do áudio fica melhor (quanto almejada a descrição fidedigna), pois leva em conta 23 ms. Obviamente o tamanho da janela FFT foi o fator diferencial.

Outra característica relevante do FFT é a decomposição da amostra sonora somente em ondas senoidais harmônicas da frequência que resulta da divisão da taxa de amostragem pelo tamanho da Janela FFT escolhida. Por exemplo, consideremos os harmônicos da

fundamental de 10.76 Hz – adquirida pela divisão da taxa de amostragem pelo tamanho da janela FFT – em uma análise com janela de 4096 e taxa de amostragem de 44100 Hz. No áudio analisado, quando há uma frequência que não é harmônica a essa "fundamental", sua amplitude será dividida entre as frequências harmônicas da fundamental que estejam mais próximas.

Por exemplo, quando uma análise FFT com essas características é realizada em um sinal de áudio que seja uma simples onda senoidal de 435 Hz, o resultado da análise produzirá picos nas amplitudes das frequências harmônicas próximas à frequência de 435 Hz (435 Hz não é harmônica à 10,76 Hz). Neste caso teremos picos nas frequências de 419.64, 430.4 e 441.16 Hz (39º, 40º e 41º harmônico de 10.76 Hz, respectivamente). Não havendo correção (que é possível, como veremos), uma ressíntese com esses dados do FFT produziria um som com efeito de *chorus* (ou até clusters se com janelas FFT menores).

Em softwares como *Spear* e *AudioSculpt* e objetos do *Max/MSP*, como o *zsa.freqpeak~*, há maneiras de se estimar a frequência aproximada. Para entendermos completamente o passo a passo devemos, primeiramente, saber que o resultado da decomposição de um som pelo FFT é uma série de números complexos, cada um representando a amplitude de um harmônico da fundamental do processo (o 20º número desta lista representa a amplitude do 20º harmônico). Essa série normalmente tem a metade do tamanho da Janela FFT utilizada, pois a outra metade é o mesmo resultado espelhado (simetria própria do cálculo), portanto, para que o algoritmo não faça cálculos desnecessários, utilizamos somente a primeira metade dos números complexos.

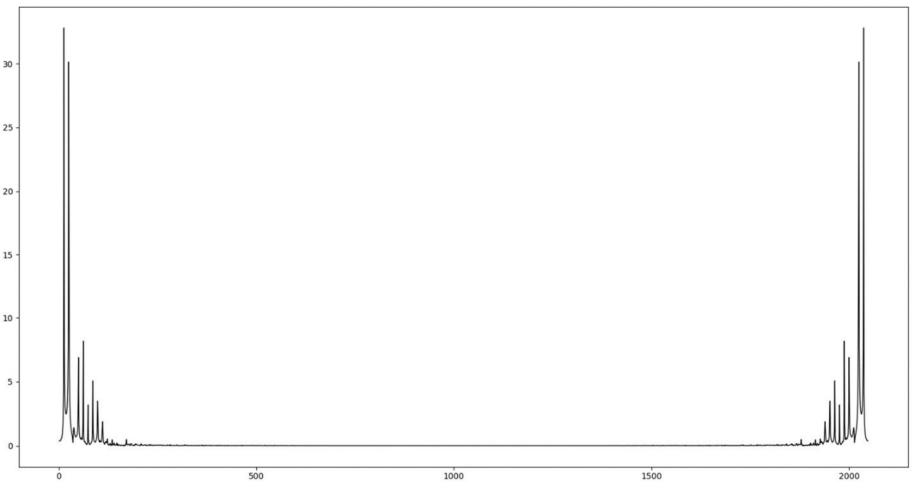

Figura 7. Demonstração da simetria própria do processo de FFT com janela FFT de 2048. Análise de um C4 de uma flauta. No qual o eixo x equivale à 2048 *bins* e eixo y equivale à amplitude de cada *bin*. Perceba a simetria reflexiva das extremidades no eixo x.

A partir desta primeira metade, para calcular a amplitude de cada *bin* (a posição de cada número complexo dentro do resultado do FFT) utilizamos a seguinte equação:

$$\sqrt{r^2 + i^2}$$

Eq. 2

Onde *r* é a parte real do número complexo e *i* é a parte imaginária do número complexo (números complexos sempre são formados por um número real e um número imaginário). A título de curiosidade, observe que a Eq. 2 é o Teorema de Pitágoras, no entanto não queremos um resultado da hipotenusa (no nosso caso a hipotenusa será a amplitude *a*) ao quadrado – $a^2 = r^2 + i^2$ –, mas sim o resultado real, por isso a raiz quadrada. Tendo calculado todas as amplitudes individualmente, podemos obter o pico aproximado de uma parábola, que corresponderá com mais precisão à frequência contida no áudio. Para isso utilizamos a fórmula sugerida por Smith (2011) (descrita com detalhes a seguir):

$$peak = \frac{1}{2}\left(\frac{a - c}{a - 2b + c}\right)$$

Eq. 3

A equação acima efetua o ajuste das frequências utilizando uma parábola criada a partir de 3 amplitudes (as variáveis *a*, *b* e *c*). Para isso é exigido que a amplitude do meio (b) (em um conjunto de 3 amplitudes) seja maior que suas duas vizinhas, condição chamada de *local maxima* (KLINGBEIL, 2009).

Observe o exemplo abaixo (figura 2), cada barra no gráfico representa a amplitude de uma determinada frequência harmônica à fundamental obtida através da divisão da Taxa de Amostragem (neste caso 44100) e o Tamanho da Janela FFT (neste caso 2048), ou seja 21,53Hz. Portanto, a primeira barra do gráfico representa a amplitude da frequência 21,53 (21,53 x 1), a segunda barra representa a frequência de 43,06 (21,53 x 2), a terceira 64,59 (21,53 x 3) e assim sucessivamente. Em nossa análise, no entanto, o que temos não é um pico representado por uma única barra, mas algumas, ou seja, inicialmente podemos pensar que no áudio há alguns clusters, porém o que o compõe são duas senoides: 440 e 960Hz, ambas com a mesma amplitude (sintetizadas no Max/MSP).

Analisando as três barras com o topo em vermelho (figura 2), podemos perceber que elas cumprem a condição de Local Maxima, portanto são utilizadas na equação 3 (eq. 3), na qual a primeira barra (da esquerda para a direita) será o valor de *a*, a segunda barra o valor de *b*, e a terceira barra o valor de *c*.

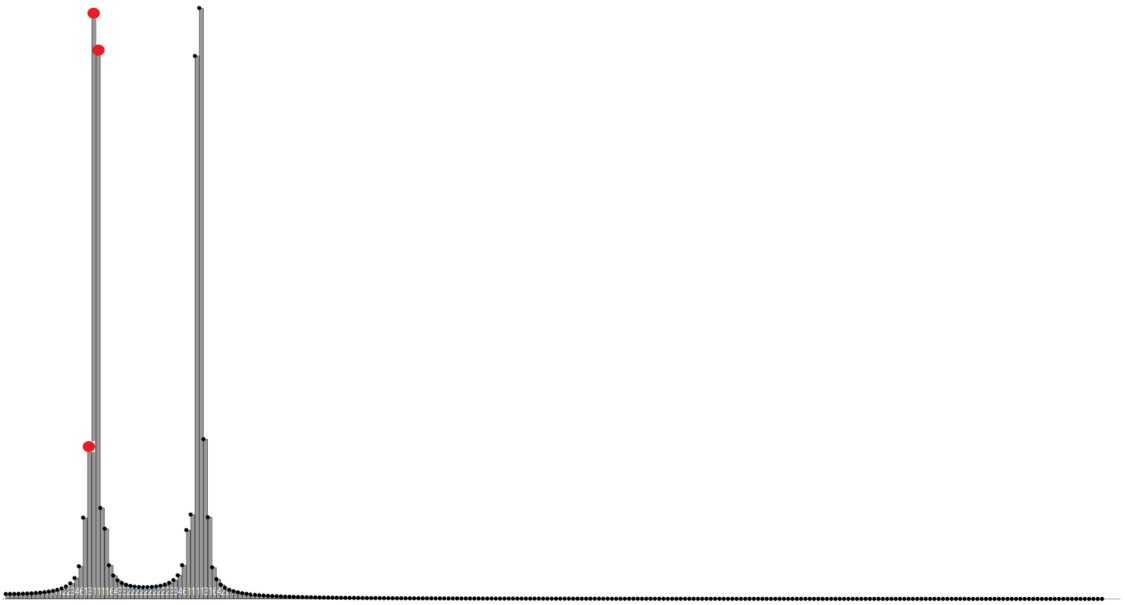

Figura 8. Exemplo de decomposição utilizando o FFT.

Neste caso específico, o valor de *a* será 31.020, de *b* 119.395 e de *c* 111.657. Sendo assim, a fórmula ficará:

$$peak = \frac{1}{2}\left(\frac{31.020 - 111.657}{31.020 - (119.395 \times 2) + 111.657}\right)$$

Eq. 4

A partir desta formula sabemos que *peak* equivale ao número 0.419. Para finalmente chegarmos à frequência contida no áudio precisamos, em sequência, somar 0.419 com a posição (index) de *b* no resultado do FFT, neste caso 119.395 (valor de b) é a 20ª amplitude na lista de resultados do FFT. Logo teremos 20 + 0.419. Finalmente, para converter o resultado para a frequência em Hertz, multiplicamos *peak* + b pela nossa fundamental 21,53 obtendo assim 439.697. Perceba que a aproximação é ótima, mas ainda não temos os 440Hz, mesmo se tratando de um som ideal dentro do axioma de onda contínua.

$$\text{Hz} = (20 + 0.419) \times \left(\frac{44100}{2048}\right) = 439.697Hz$$

Eq. 5

A partir desse exemplo podemos perceber que os processos de FFT usados por alguns dos mais populares softwares, como o *Spear* e o *AudioSculpt*, fazem estimativas aproximadas das frequências espectrais que estão, portanto, sujeitas a imprecisões,

principalmente quando usamos parâmetros automatizados. Logo, composições nas quais o material musical é um timbre a partir do qual se busca criar uma melodia/gesto através do caminho de um parcial, a variação dos parâmetros de janela FFT tem influência direta no resultado musical da composição. Problematizando a partir de outra perspectiva, uma vez que se utiliza o Spear para fazer a modelagem espectral dos parciais (*partial-tracking*), devemos entender que o que faz o processo é um *modelo* matemático/algorítmico que faz *uma* descrição do áudio, portanto, nas aproximações espectrais que fazem uso de melodias de parciais, por exemplo, o que temos é o uso de *um* modelo matemático que se aproxima do timbre, não o timbre em si mesmo, não uma descrição totalmente fidedigna ao timbre – ressaltando que essa descrição será influenciada pela escolha dos parâmetros por parte do(a) artista. O Spear, por exemplo, automatiza um parâmetro importantíssimo para essa descrição que é o limite máximo que um parcial pode ser alterado para ser considerado o mesmo parcial. Portanto, o que entendemos como problemático são duas questões: a falta de consciência da interferência dos algoritmos e a automatização dos parâmetros do algoritmo, que apesar de facilitar o uso por parte de compositores(as), podem impor escolhas/decisões estéticas de quem os programou.

Concordamos, portanto, com o conceito de “orquestração eletroacústica” (THOMASI, 2016), principalmente ao modo refletido por Ribeiro (2018) e Roads (2015) que aproximam os parâmetros envolvidos na música eletroacústica da orquestração. Em nossa concepção, podemos conhecer as características de nossos instrumentos (leia-se softwares) eletroacústicos e utilizá-las expressivamente, ou não os conhecer e muitas vezes deixar suas marcas/características se imporem em nossas obras, ao final de tudo temos uma escolha estética.

3. O PROCESSO DE FAST FOURIER TRANSFORM NO OPENMUSIC

Entendendo que é necessário compreender algumas das características do FFT para dar alguma escolha aos(às) compositores(as), a seguir, descrevemos brevemente uma implementação no *OpenMusic* que busca servir como ferramenta didática para testes e para a compreensão prática das características apontadas acima. Apresentamos resultados parciais da implementação que está em processo. Novamente salientamos que o intuito desta biblioteca é servir como ferramenta didática, uma vez que possibilita ver o código da

implementação, desta forma entender o passo a passo do processo (lembrando que isso não ocorre na OM-SuperVP e OM-pm2 por serem proprietárias e serem entregues em binários).

Acreditamos que um dos benefícios de implementar FFT no *OpenMusic* é demonstrar visualmente algumas questões, ao mesmo tempo em que ele nos permite analisar o código por detrás dos objetos, ainda que ao preço de ter um processamento mais lento. Faz parte de nosso objetivo comentar os códigos, para torná-lo acessível a artistas/estudantes que não saibam programar, como é ilustrado pelo exemplo da Figura 3.

Podemos entender, por exemplo, como cada uma das funções de janela (*window-functions*), utilizadas para melhorar a descrição STFT, modificam a onda sonora (vide Figura 4) e, ao mesmo tempo, comparar como esses diferentes tipos de janelas trazem diferentes resultados para o processamento FFT (vide Tabela 1).

```
(defun sound-window (sound-bytes-window window hop-size windows-type &optional result)

(let* (
      (action1 (if (equal nil windows-type) ;; checa se o(a) usuario selecionou ou nao um tipo de janela para a analise.

                (list-to-array (first-n sound-bytes-window window) 1) ; se nao selecionou, o algoritmo simpleste
                                                                      ; organiza os samples de acordo com a hop-size e
                                                                      ; os transforma em arrays (estruturas de dados que
                                                                      ; agilizam o processamento).

                ;; First-n seleciona os primeiros n bytes de todos os bytes do sample(audio) em analise. Sendo que n e o hopsize.

                (om-ckn::apply-window (list-to-array (first-n sound-bytes-window window) 1) windows-type)))
                                                        ; se o(a) usuario(a) selecionou, organizamos os samples
                                                        ; e entao aplicamos o calculo da janela atraves da funcao
                                                        ; om-ckn::apply-window

;; ============================================================================================

      (action2 (last-n sound-bytes-window (let* ((number (- (length sound-bytes-window) hop-size)))
                                            (if (plusp number) number 1)))))

                ;; Esta acao exclui os samples que ja passaram pelo organizacao e a transformacao para arrays.
                ;; Em seguida checa se ha samples suficientes para fazer o calculo da ACTION1 novamente.

;; ============================================================================================


(if (< (length (remove nil action2)) window) ;; Esta acao checa o resultado da ACTION2 e ve se o algoritmo deve finalizar o calculo
                                             ;; ou fazer a ACTION1 e ACTION2 novamente.

;; ============
    (reverse (om::x-append (list action1) result)) ;; Nao havendo mais samples para organizar e fazer os calculos, aqui recolhemos todos
                                                   ;; os resultados e direcionamos para o output.
;; ============

    (setf sound-bytes-window (sound-window action2 window hop-size windows-type (push action1 result))))))
                                                   ;; Caso seja necessario fazer o calculo novamente aqui salvamos os resultados na
                                                   ;; na variavel RESULT e entao fazemos o calculo novamente.
```

Figura 9. Exemplo de código em OpenMusic que determina o *window-size* e organiza os samples.

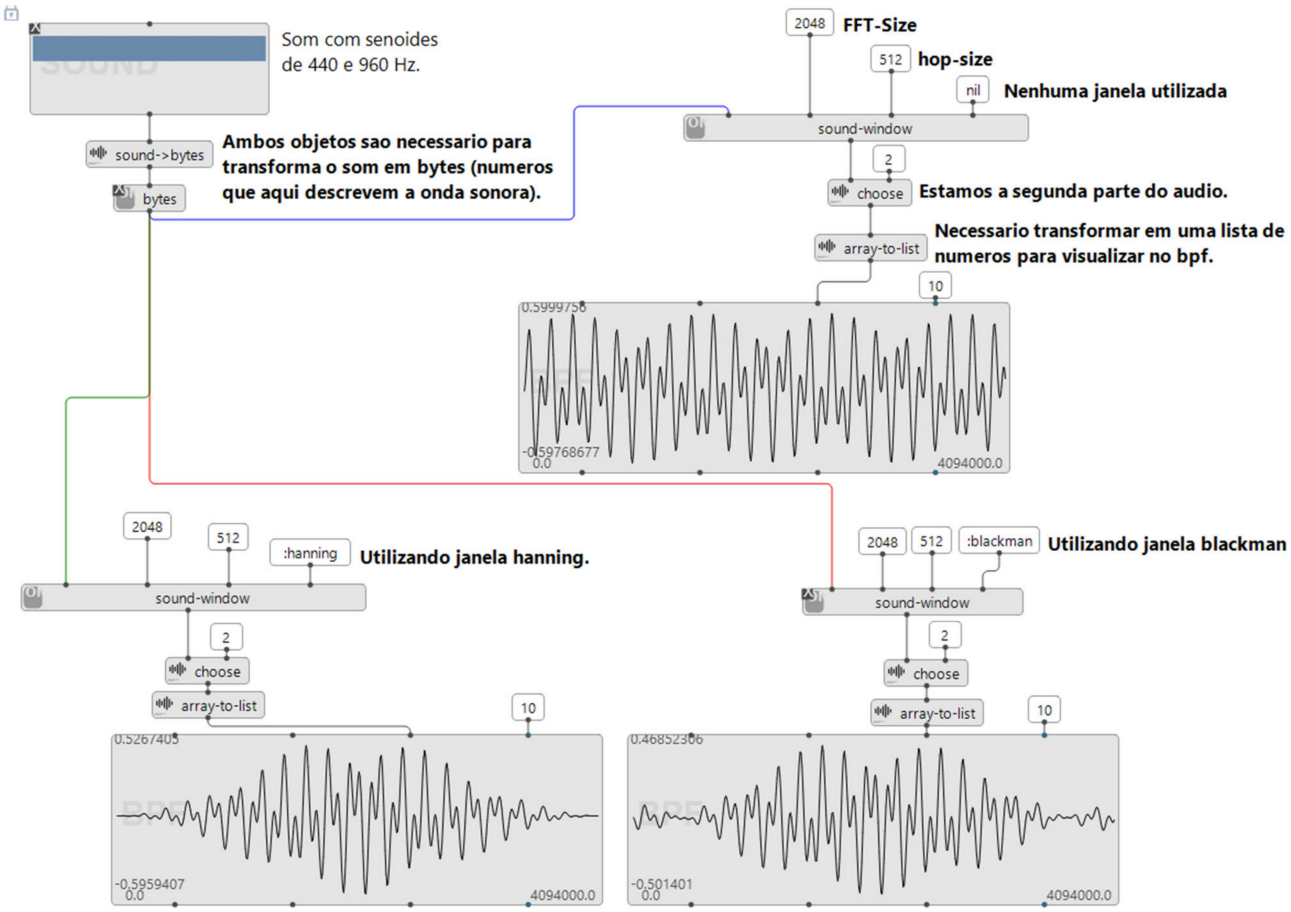

Figura 10. Exemplo do usa da onda sonora com window-functions para minimizar problemas do STFT.

Tabela 1. Comparação da diferença entre janelas utilizadas no processo de FFT.

Janela Utilizada	Som original	Resultado com filtro de -60dB
Hanning	440 0.3 590 0.1	440 0.36 590 0.12
Blackman	440 0.3 590 0.1	440 0.32 592 0.12
Barlet	440 0.3 590 0.1	440 0.37 590 0.12 288 0.0017 244 0.0011
Nenhuma	440 0.3 590 0.1	437 0.55 587 0.18 2337 0.0022 3228 0.0015

Note o motivo das maiorias dos softwares utilizarem a janela *Hanning* (Max/MSP e PureData) ou a *Blackman* (Spear). Elas oferecem uma ótima precisão na descrição das frequências. No entanto, podemos ver que na comparação das duas janelas, a *Blackman* se sai melhor na descrição das amplitudes a janela *Hanning* é melhor na descrição das frequências (lembrando que estamos lidando com um ambiente ideal no qual temos somente duas senoides).

A partir dessa breve demonstração, realizada com duas ondas senoidais, podemos levantar as seguintes questões: Qual é o nível de distorção de um processamento em tempo real que utilize FFT em samples gravados ruidosos? O(A) artista percebe e tem consciência dessas distorções? Como avaliar tais impactos esteticamente? Ele(a) compreende que a escolha da janela FFT pode afetar diretamente questões da prática musical?

A biblioteca, os processos e os help patches referentes a esta pesquisa podem ser encontrados no endereço https://github.com/charlesneimog/OM-CKN/releases/. Optamos por utilizar nela o software livre OM-Sharp (desenvolvido a partir do OpenMusic) que está disponível para todas as plataformas no endereço https://github.com/cac-t-u-s/om-sharp/releases/.

4. PRÓXIMOS PASSOS DESTE PROJETO DE FAST FOURIER TRANSFORM NO OPENMUSIC

Futuramente desenvolveremos formas de medir resultados aplicados a diferentes tipos de janelas, procurando entender as características do processo de descrição de áudio em diferentes formas de aplicação do STFT. Nos interessam questões como: Qual o impacto do uso de diferentes *window-functions* na modelagem senoidal? Há diferenças de processos na descrição de áudio em softwares como o Spear (em tempo diferido) e softwares como o Max/MSP e PureData (em tempo real)?

Após essa etapa, nosso intuito será investigar quais são os problemas composicionais ao usar essas ferramentas sem um conhecimento técnico mínimo, buscando responder questões como: help patches e externos (com seus códigos compilados) teriam a capacidade de modificar e/ou propor estruturas de composição eletroacústica a artistas que não tem conhecimento da lógica desses processos? Até que ponto isso pode ser entendido como um processo de aculturação por quem constrói esses softwares? Em que medida podemos avaliar esse possível impacto em nosso processo criativo sem nos tornarmos programadores?

5. REFERÊNCIAS

CAESAR, Rodolfo. **O enigma de lupe.** Pequena Biblioteca de Ensaios. Rio de Janeiro: Zazie, 2016.

CAMPOS JÚNIOR, José Ignácio de. **Interação Tímbrica na Música Eletroacústica Mista.** Dissertação (Mestrado em Música) – Instituto de Artes, UNICAMP, Campinas, 2005.

FLUSSER, Vilém. **Filosofia da Caixa Preta: ensaios para uma futura filosofia da fotografia.** Rio de Janeiro: É Realizações Editora, 2018.

KLINGBEIL, Michael Kateley. **Spectral Analysis, Editing, and Resynthesis: Methods and Applications. Orientador: Tristan Murail.** (Doutorado em Música) – Graduate School of Arts and Sciences, Columbia University. New York. 2009.

LOY, Gareth**. Musimathics: The Mathematical Foundations of Music**. v. II. Cambridge: MIT Press, 2007.

RIBEIRO, Felipe de Almeida. **O impacto dos sintetizadores no processo composicional.** Opus, v. 24, n. 1, p. 167-186, jan/abr. 2018.

ROADS, Curtis. **Composing Electronic Music - A New Aesthetic.** New York: Oxford University Press, 2015.

SMITH, Julius O. **Spectral Audio Signal Processing.** W3K Publishing: Stanford University. 2011.

THOMASI, Ricardo Oliveira. A função multidisciplinar do compositor eletroacústico: uma abordagem operacional. **Revista Vórtex,** v. 4, n. 2, p. 1-9, 2016.