API para obtenção da Frequência de um Sinal de Som por meio da FFT em Java ME

Marcelo Ruaro

Departamento de Engenharias e Ciência da Computação – Universidade Regional Integrada do Alto Uruguai e das Missões (URI)
Caixa Postal 98.802-407 – Santo Ângelo – RS – Brasil

mcruaro@gmail.com

Resumo. *Este artigo expõe o desenvolvimento, na linguagem Java Micro Edition, da API FastFrequency, capaz de prover a obtenção da frequência de um sinal de som utilizando a Transformada Rápida de Fourier(FFT) implementada pelo algoritmo radix-2 posteriormente ao janelamento do sinal através das janelas de Hamming. Estas frequências podem ser utilizadas tanto como um produto final, como também, uma fonte para a aplicação de metodologias para um processamento de espectro mais robusto, principalmente exemplificado pelas técnicas que compõe o estado da arte do Reconhecimento de Voz.*

1. Introdução

A Transformada Rápida de Fourier (FFT) ao longo dos anos foi fundamental para o desenvolvimento da comunicação digital. Essa técnica permite a obtenção da frequência física de um determinado sinal, retornando como saída, valores que podem ser utilizados em diversos cenários do processamento de sinais digitais, como o cálculo do espectro de potências e fase [1]. No reconhecimento de voz, ela é amplamente utilizada, e deriva a aplicação de técnicas posteriores, como os coeficientes LPC, LPC-Cepstrais, Mel-Cepstrais, entre outros [2].

O cálculo da FFT sobre um sinal é realizado posteriormente a aplicação da técnica de janelamento ao sinal pré-processado, o que permite segmentar um sinal de tempo x em pequenas frações, geralmente sobrepostas, que são consideradas estacionárias em relação a sua variação no tempo, permitindo o uso de técnicas futuras para extração de informação do sinal [6].

Introduzida estas técnicas, este trabalho tem como objetivo apresentar o desenvolvimento da API *FastFrequency*, cuja capacidade é o de obter um conjunto de frequências originais de um determinado intervalo de som. Com a *FastFrequency*, desenvolvedores e pesquisadores poderão abstrair fases antecedentes ao processamento de espectro após a FFT, e iniciar um desenvolvimento mais focado em metodologias de maior complexidade, como as que abrangem o reconhecimento de voz [3] [6]. Também, o vetor de frequência gerado, poderá ser utilizado como um produto final em aplicações que requisitem a obtenção de frequências do sinal, como por exemplo, um afinador musical [4].

Para o desenvolvimento da API e consequentemente obtenção das frequências foi utilizada a FFT, implementada pelo algoritmo *radix-2* adaptado para linguagem *Java*

Micro Edition, com aritmética de ponto flutuante (*float*), sobre um sinal janelado pelas janelas de *Hamming*.

Na sessão 2 a seguir, será descrito o desenvolvimento e implementação. Na sessão 3 será exposta a estrutura da API, a sessão 4 apresenta os resultados, e a sessão 5 encerra, expondo as conclusões.

2. Desenvolvimento e Implementação

A maior parte desta sessão é voltada a obtenção da frequência por meio dos processos que envolvem a FFT. Porém primeiramente, é conveniente expor uma contextualização sobre o método de captura do som através do dispositivo móvel na linguagem Java ME.

2.1. Captura do Som

A captura foi realizada através da interface de programação de aplicações (API) MMAPI Goyal (2006), na modulação (*Pulse Code Modulation*) PCM no formato *Wave* com uma frequência de amostragem de no mínimo 8KHz, podendo variar para valores maiores que este, caso necessário uma representação maior que 4KHz, segundo o teorema de *Nyquist*. Já a quantidades de bits por amostra é fixado em 8 bits constituindo um vetor de *bytes* que apresentará variações de amplitude de -128 a 127.

2.2. Pré-processamento

O sinal em seu formato original foi submetido a um pré-processamento a fim de eliminar seu cabeçalho *Wave* removendo as primeiras 56 posições do vetor de *bytes* que caracteriza a amostra de som [4].

Outro processo adicional, que pode ser empregado, é a remoção dos períodos de silêncio da elocução [3], este processo é disponibilizado na API, mas esta técnica não foi utilizada para obtenção do vetor de frequências, pois constatou-se que os períodos de silêncio são facilmente detectados, resultando em frequências com valor de 0 Hz.

2.3. Janelamento

Com a aplicação das janelas de *Hamming* a um tamanho inicial de 256 amostras e sobreposição de 40%, foi possível segmentar o sinal em partes de curta duração [5]. Esta segmentação permitiu extrair uma frequência única para cada janela, pois um sinal de som em sua totalidade (principalmente o do trato vocal), geralmente apresenta uma gama complexa de frequências [6], sendo que não se pode adotar uma única frequência como representante do sinal.

2.4. Aplicação da FFT

Em seguinda de posse das janelas foi aplicado o cálculo da FFT. A FFT corresponde a uma família de algoritmos que implementam de uma forma mais rápida a Transformada Discreta de *Fourier* e permite alterar a representação de um sinal do domínio de tempo para um domínio de frequências [1].

Neste trabalho a FFT foi adaptada para a linguagem Java ME, com base no algoritmo *radix-2*, recebendo como entrada uma parte real e outra imaginária, sendo que ambas devem ter uma tamanho igual a potência de dois. Como parte real, foi atribuído o sinal janelado, e para a parte imaginária, foi atribuído zero [7].

Após a aplicação da FFT obteve-se o espetro de magnitudes correspondendo a representação no domínio de frequência do sinal de entrada. Com isto, tem-se um vetor (para cada janela), onde cada índice representa uma escala de frequências baseada na faixa de frequência de amostragem definida na captura, e cada elemento representa a magnitude de cada frequência (índice), presente no sinal. Deste modo, pode-se identificar a frequência através da procura pela maior amplitude do vetor e associar o índice em que esta se encontra como a frequência da janela analisada. Logo para transformar o índice em uma frequência do sinal, foi proposta a seguinte equação baseada no algoritmo apresentado por Heart *et al* (2007):

$$f = si/\left[2\left(\frac{N}{2} - 1\right)\right]$$

onde f é a frequência, s é a frequência de amostragem, i é o índice que contém a maior amplitude do vetor de magnitudes e N é o número de pontos da FFT, logo N é o igual ao número de amostras por janela. Desta maneira, obteve-se um vetor caracterizando as frequências cálculadas do sinal de entrada ao longo do tempo.

3. Estrutura da API

Nesta sessão busca-se descrever a API *FastFrequency* de uma forma prática e clara. Inicialmente, a figura 1 abaixo explica seu contexto, onde se identifica a característica de infraestrutura da API, permitindo o desenvolvimento de diversas aplicações que requisitem um processamento de espectro.

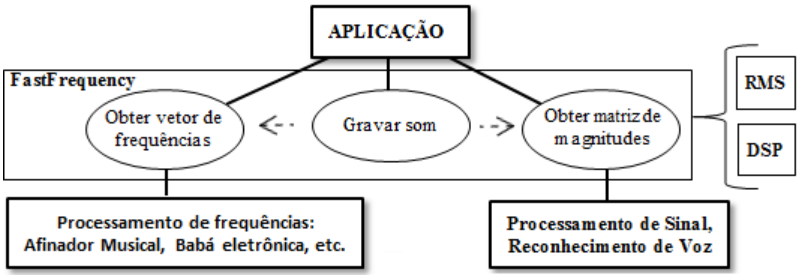

Figura 1. Contexto da API

Na figura 1, percebe-se que a API possui a capacidade de retornar unicamente a matriz de magnitudes. Esta matriz corresponde à saída original da FFT para cada janela sem a aplicação da equação apresentada acima [1]. Onde, a partir desta matriz, obtém-se o espetro de potências, que pode ser empregado em metodologias de processamento de espectro mais robusto, como por exemplo, o reconhecimento de voz [8], reconhecimento de som [4], ou outras técnicas de processamento de sinais digitais [3].

3.1. Utilização

A API é de fácil entendimento, e através da instanciação das classes *Sample* e *FastFrequency*, permite ao desenvolvedor utilizar poucas linhas de código para a obtenção do vetor de frequências. A seguir são descritas estas classes, juntamente com seus principais métodos.

A classe ***Sample*** permite a gravação de uma amostra para ser inserida no processo de obtenção de frequências. Possui sobrecarga de construtor, permitindo a passagem do tempo total de gravação (*time*), ou este tempo, acompanhado da frequência de amostragem (*sampleRate*), que deve ser no mínimo de 8KHz.

A classe ***FastFrequency*** é a principal classe da API, e em seu construtor recebe um objeto do tipo *Sample* contendo a amostra de som gravada. Também possui sobrecarga de contrutor permitindo a passagem de mais dois parâmetros, que são *windowsWidth*, correspondente a largura das janelas de *Hamming*, e *overlap* correpondendo a sobreposição entre as janelas [3]. Seu principal método é o *getFrequency* – ver figura 2, que retorna um vetor contendo uma série de frequências referente a segmentos do sinal ao longo do tempo. Esta classe também fornece o método *getFFToutput*, que retorna a matriz de magnitudes citada acima.

```
import fastFrequencyAPI.*;

Sample sample = new Sample(2000); //grava por 2 segundos
FastFrequency fastFreq = new FastFrequency(sample);
floatArray = fastFreq.getFrequency(); //retorna as frequências
```

Figura 2. Exemplo de obtenção de frequência em Java ME

Para utilização da API, é necessário adicionar seu *.jar* ao projeto, e realizar um *import* na classe que deseja utiliza-la através da diretriz: *fastFrequencyAPI.**.

4. Testes e Resultados

Para a aquisição das frequências certos parâmetros são fundamentais, como o número de amostras por janela e a frequência de amostragem. O número de amostras por janela caracteriza o tamanho da janela, e neste trabalho foi caracterizado com o requisito obrigatório do mesmo ser uma potência de dois, devido a sua inserção no algoritmo *radix-2*. Já a frequência de amostragem, foi adaptada conforme a necessidade de se obter frequências mais elevadas, respeitando o teorema de *Nyquist*.

Abaixo são expostos os resultados obtidos, destacando determinadas frequências, e as relacionando com a frequência obtida, juntamente com o tempo gasto de todo o processo de extração de frequência para um sinal de dois segundos em um dispositivo móvel, com processador operando a frequência de 464MHz, e acessando a aplicação de sua própria memória interna.

Tabela 1. Frequências obtidas em Hz para cada tamanho de janela, em uma frequência de amostragem de 8000Hz

Freq. Original	Freq. obtida para cada tamanho de janela			
	256	***512***	***1024***	***2048***
50	62	47	46	50
60	62	62	62	58
100	94	94	101	101
200	188	203	203	199
300	314	298	297	301
500	503	501	500	500
1000	1007	1003	1001	1000
1500	1511	1505	1502	1501
2000	2015	2007	2003	2001
3000	3023	3011	3005	3002
4000	4000	4000	4000	4000
Tempo Médio	**1,0s**	**1,15s**	**1,26s**	**1,35s**

Tabela 2. Frequência obtida em Hz , para cada tamanho de janela em uma frequência de amostragem de 16000Hz

Freq. Original	Freq. obtida para cada tamanho de janela			
	256	**512**	**1024**	**2048**
8000	8000	8000	8000	8000
Tempo Médio	**2,15s**	**2,25s**	**2,5s**	**2,75**

Com base nos resultados pode-se perceber o melhor detalhamento de cada frequência ao se elevar o tamanho das janelas. Isso ocorreu, pois uma janela maior oferece uma maior faixa de representação de frequências sobre uma janela menor, entretanto, a probabilidade de rápidas variações de frequências serem suprimidas é acrescida. Também, verifica-se que os tempos crescem proporcionalmente ao tamanho das janelas e frequência de amostragem.

Contudo, percebeu-se que janelas menores limitam uma representação mais precisa de frequências justamente por conter menos índices. Seguindo esta linha, na tabela 3 a seguir é exposto o nível de detalhamento que cada tamanho de janela oferece para representação de frequências.

Tabela 3. Espaços na representação de frequência de acordo com o tamanho de janela

Amostras por Janela	8000Hz
128	62,5Hz
256	31,25Hz
512	15,62Hz
1024	7,8125Hz
2048	3,90625Hz

Este cálculo pode ser efetuado dividindo a metade de determinada frequência de amostragem (*Teorema de Nyquist*) por metade do tamanho original das janelas [8]. Como resultado obtêm-se os valores apresentados na tabela 3 acima, onde quanto menor o índice mais fiel será a representação das frequências capturadas, pois maior será a possibilidade de representação – ver figura 3.

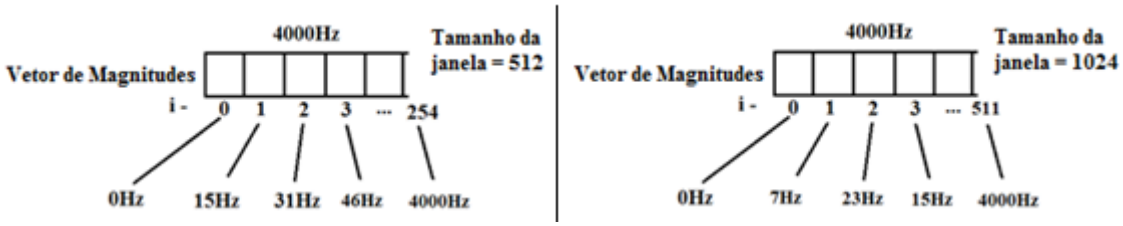

Figura 3. Representação de frequência em diferentes tamanhos de janelas

Ou seja, o detalhe de representação das frequências conterá saltos, por exemplo, de 15,6 hertz para um tamanho de janela igual a 512 amostras, 7,8 hertz para janelas de 1024, e assim sucessivamente.

5. Conclusão

Este trabalho apresentou uma API para obtenção da frequência aproximada por meio da Transformada Rápida de *Fourier* para a linguagem *Java Micro Edition*.

Com base nos resultados apresentado ao longo das pesquisas conclui-se que as frequências obtidas encontram-se em uma faixa, senão igual, muito próxima a

frequência original, onde ainda pode-se considerar ruídos causados pela qualidade do material do transdutor, ou interferências específicas de cada ambiente [6].

Outro ponto observado foi que, para um tamanho de janela menor, por exemplo, 256 amostras, o espaço de representação de frequências é reduzida pela metade, pois segundo Cuadros (2007) o vetor de magnitudes resultante das funções seno e cosseno da FFT tem suas metades simétricas, logo descarta-se a segunda metade pois os valores serão iguais ao da primeira metade. Desta maneira, o vetor processado conterá – para uma frequência de amostragem de 8000Hz, por exemplo – uma gama de representação sobre 4000Hz de acordo com o teorema de *Nyquist*, e variando a cada 31,25 Hz, pois sua faixa de representação está limitada em 128 variações (256 / 2). Este conceito segue-se para as demais frequências de amostragem e tamanho de janelas, onde, quanto maior for a janela, maior será a faixa de representação de frequências, porém maiores as chances de curtíssimas alterações de frequências do sinal serem suprimidas.

Por fim, conclui-se que a API desenvolvida permite uma prática, mas também robusta implementação das funcionalidade disponibilizadas, permitindo seu uso pelos mais variados tipos de desenvolvedores, desde estudantes a profissionais avançados.

Referências

1. CERNA, M.; HARVEY, A. F. (2000)"The Fundamentals of FFT-Based Signal Analysis and Measurement". National Instruments, Junho.

2. LIMA, A. A. D.(2000) "Análises Comparativas em Sistemas de Reconhecimento de Voz". Universidade Regional do Rio de Janeiro, Rio de Janeiro, Setembro.

4. RUARO, M. SRM: (2010)"Framework para Reconhecimento de Som em Dispositivos Móveis. Universidade Regional Integrada do Alto Uruguai e das Missões", Santo Ângelo, Dezembro. 91.

5. HERATH, I.; RAGEL, R. G.(2007) "Implementation of an Electronic Tuner in J2ME using Fast Fourier Transform". Peradeniya University Research Sessions, Sri Lanka, Vol.12, Part II, Peradeniya, 30 Novembro.

6. RABINER, L.; JUANG, B.-H. (1978)"Fundamentals of Speech Recognition". New Jersey: Englewood Cliffs: Prentice Hall.

7. OLIVEIRA, K. M. D. "Reconhecimento de Voz Através do Reconhecimento de Padrões". Universidade Católica de Salvador - UCSAL, Salvador, Dezembro 2000.

8. CUADROS, C. D. R. (2007)"Reconhecimento De Voz E De Locutor Em Ambientes Ruidosos: Comparacão Das Técnicas Mfcc E Zcpa". Escola de Engenharia da Universidade Federal Fluminense, Niterói.

9. PETRY, A.(2002) "Reconhecimento Automático de Locutor Utilizando Medidas de Invariantes Dinâmicas Não-Lineares". URGS, Inst. de Informática, Porto Alegre, Agosto.