Sistema Especialista Para Reconhecimento De Acordes Musicais Em Tempo Real Para Violão Elétrico Utilizando Técnicas De DSP

Sandro Alex de Souza Ferreira

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Bahia, como parte dos requisitos para obtenção do grau de Mestre em Engenharia Elétrica.

Área de Concentração: Processamento de Sinais

Antonio Cezar de Castro Lima, Ph.D.
Orientador

Sistema Especialista Para Reconhecimento De Acordes Musicais Em Tempo Real Para Violão Elétrico Utilizando Técnicas De DSP

Sandro Alex de Souza Ferreira

Dissertação de Mestrado

Antonio Cezar de Castro Lima, Ph.D.
Orientador

Luiz Alberto Luz de Almeida, Ph.D.
UFBA
Edson Costa de Barros Carvalho Filho, Ph.D.
UFPE

Salvador, Bahia, Brasil Dezembro de 2006

Dedicatória

Ao senhor Wilson (meu pai) e dona Adineia (minha mãe) que sempre foram compreensivos e pacientes me apoiando em todos os meus projetos e que tanto me estimularam para alcançar mais esta conquista.

Agradecimentos

Agradeço a Deus, em primeiro lugar, por ter me dado uma vida, saúde e família maravilhosas; a meus pais que sempre acreditaram, confiaram e que me encaminharam com suas experiências de vida; a Meiriana por ser muito mais que uma companheira, ajudando sempre em períodos complicados; a Acácia Gomes, chefe imediata, pela compreensão e apoio efetivo para conciliar os meus estudos ao horário de trabalho; a Gabriel que fez uma pesquisa semelhante; a Paulo Cortês pelas idéias interessantes; aos professores Dr. Eduardo Telmo – CEFET-Ba, Dr. Jamary Oliveira – Música/UFBA, Dr. Pedro Kröger – Música/UFBA, Dr. Ricardo Bordini – Música/UFBA e PhD. Antonio Cezar de Castro Lima (orientador) que com seus direcionamentos e sabedoria contribuíram para a conclusão deste trabalho; a Irina Santiago por sua análise crítica e construtiva sobre o texto e as imagens; aos meus amigos por compreenderem minha ausência enquanto me dedicava na realização deste trabalho; e a todos aqueles que contribuíram de alguma forma para que esse projeto obtivesse sucesso.

Resumo

Este estudo exploratório busca desvendar a complexidade da interação entre a física, a música e o computador, e apresenta um software capaz de realizar o reconhecimento de acordes musicais maiores e menores do violão elétrico em tempo real. Para atingir os resultados desejados, foram utilizadas a transformada rápida de Fourier - FFT e outras técnicas de processamento digital de sinais - DSP, além de sistemas especialistas para detecção e classificação de acordes maiores e menores. A aplicação foi desenvolvida e testada utilizando o ambiente MatLab, o qual atingiu o resultado esperado academicamente, podendo adquirir aprimoramentos para atingir requisitos comerciais. As reflexões e o desenvolvimento do projeto levaram à conclusão de que é realmente possível identificar acordes maiores e menores do violão elétrico em tempo real utilizando-se técnicas de DSP.

Palavras-chave: Processamento digital de sinais, transformada rápida de Fourier, reconhecimento de acordes, músicas, sistemas especialistas.

Abstract

This exploratory study unveils the complexity of the interaction among Physics, music and Computer science. This work presents a software capable of recognize the major and minor musical chords from the electric guitar in real time. To achieve these results, it was sender a spectral analyzes based on Fast Fourier Transform - FFT and other techniques of Digital Signal Processing. Moreover an expert system, for detection and classification of major and minor chords, was employed. The program was developed and tested using the MatLab environment in order to reached the expected academic results. Its commercial application could be easily implemented with few developments in C language. The reflections and the development of the project had led to the conclusion that it is really possible to identify major and minor chords.

Key Words: Digital Signal Processing, Fast Fourier Transform, chord recognition, music, expert system.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Os computadores estão cada vez mais substituindo os seres humanos na realização de tarefas que agora podem ser realizadas de forma repetitiva e eficaz, por meio de suas grandes capacidades de processamento e memória. As funções sensoriais, entretanto, tais como paladar, audição e visão ainda estão muito longe de serem fielmente exercida por eles [Ste96]. Esta dissertação busca descrever os possíveis métodos e aplicações da tecnologia da informação na música, buscando correlacionar conceitos e teorias importantes para a integração da música e suas teorias ao processamento digital de sinais e seus processos. A partir daí, foi criado um Sistema de Reconhecimento capaz de identificar acordes maiores e menores em tempo real para violão elétrico.

A transcrição musical automática tem atraído o interesse de músicos e cientistas da computação por mais de 25 anos [Kla04] [MQ05]. Transcrever um trecho de música pode ser definido como o ato de escutar e escrever o conteúdo correspondente ao que se ouviu [Nag03]. A motivação para o desenvolvimento deste trabalho está na necessidade da criação de um dos módulos (altura[1] das notas) do Sistema de Reconhecimento de acordes musicais para instrumentos reais capaz de analisar o ritmo, a altura das notas e os instrumentos. Atualmente existe uma vasta gama de aplicativos que trabalham somente com instrumentos sintetizados ou com melodia monofônica[2] e fazem a tradução para o Musical Instrument Digital Interface – MIDI [Mac01].

Os problemas computacionais correspondentes a todas as análises da percepção computacional da música podem ser resumidos à detecção do ritmo, da altura das notas e à análise dos instrumentos musicais [Nag03]. Um dos grandes problemas no Sistema de Reconhecimento é apresentar os acordes identificados no mesmo instante de tempo em que são tocados no instrumento. A execução do aplicativo consiste em resolver uma série de funções matemáticas,

[1] Altura: em música, altura refere-se à forma como o ouvido humano percebe a freqüência dos sons.

[2] Monofonia: duas ou mais notas nunca soam simultaneamente.

de custo computacional relativamente alto, em amostras de áudio digitalizadas e identificar a altura de cada sinal para poder associar a uma conclusão baseada em conhecimentos musicais, sendo que estes sinais podem ter tempo e duração diferetes.

Devido a estas e outras dificuldades encontradas, foram definidas algumas restrições nesta dissertação a fim de tornar possível a elaboração de um trabalho conclusivo sobre o tema. Foi adotada a cifra como notação musical, pois considera apenas a altura das notas. Outras informações tais como unidades de tempo, compasso, andamento e demais parâmetros musicais serão ignorados. As análises de tipos de instrumentos musicais (fontes sonoras) diferentes e detecção de ritmo também serão ignoradas. O violão elétrico foi adotado como fonte sonora por ser um dos instrumentos musicais mais populares, relativamente fácil de transportar se comparado com um piano, por exemplo, e não necessita de energia elétrica para seu funcionamento. Dada a complexidade destas questões, foi preferido restringir a pesquisa, deixando estes pontos que não foram analisados para outros estudos e pesquisas.

O estudo teve por objetivo desenvolver um aplicativo capaz de reconhecer acordes musicais maiores e menores do violão elétrico em tempo real (conectado à entrada de áudio da placa de som em um computador), utilizando a Transformada Rápida de Fourier (FFT), técnicas de processamento digital de sinais – DSP, além de Sistemas Especialistas para detecção e classificação dos acordes. Este aplicativo poderá servir no futuro como um dos módulos para um Sistema de Reconhecimento de acordes musicais mais complexo que seja capaz de reconhecer outros tipos de instrumentos musicais e que inclua outros parâmetros musicais aqui desconsiderados. Além disso, espera-se contribuir em pesquisas na área de percepção computacional da música, além de poder auxiliar outros projetos que envolvam tópicos aqui relacionados tais como teoria musical, processamento digital de sinais, matemática e a música, dentre outros.

Como se trata de um estudo de caráter exploratório, foi utilizado como metodologia a pesquisa bibliográfica em fontes secundárias para sua fundamentação teórica. Esta dissertação está dividida da seguinte forma: primeiro são apresentadas características físicas do som (capítulo 2), para definir importantes conceitos na compreensão do trabalho como, por exemplo, intensidade, timbre, freqüência e a percepção do som. A seguir é mostrada a distribuição de faixas de freqüências, os intervalos de freqüências que são destinadas às notas musicais (oitavas), as escalas musicais e como calcular os intervalos entre notas musicais no sistema igualmente dividido.

O capítulo sobre teoria musical (capítulo 3) mostra uma visão geral sobre os conceitos da música. Explica ainda a padronização das freqüências sonoras, as notas e freqüências no violão, diferencia os acordes musicais maiores e menores, além de exemplificar a sua utilização no violão.

O capítulo 4 explica, passo a passo, como foi possível obter sucesso com a implementação do Sistema de Reconhecimento, aborda os pontos mais complexos, indica outras possíveis maneiras de se chegar ao resultado obtido e define também a estrutura utilizada para a construção do mesmo. No capítulo 5 são mostrados os resultados dos testes realizados para validação do Sistema. Por fim, a conclusão (capítulo 6) ressalta os pontos fortes e as limitações do estudo, as dificuldades encontradas para a sua realização e sugestões para trabalhos futuros.

Capítulo 2

A Física da Música

Quando a palavra música é citada, raramente se pensa em física ou na análise científica necessária para entender os processos que antecedem a apreciação da arte musical. A relação entre a física e a música começou a aparecer depois da teoria ondulatória, estabelecida nos séculos XVII e XVIII, e sedimentou-se quando Jean-Baptiste Joseph Fourier desenvolveu no inicio do século XIX a formulação da matemática, que leva seu sobrenome, e que é utilizada na análise de qualquer fenômeno periódico. Sendo a música a arte dos sons, é intuitivo associar-lhe à importância dos fenômenos físicos presentes na produção, transmissão e recepção do mesmo [Med97].

2.1 Propriedades Físicas do Som

O som é uma compressão mecânica ou onda longitudinal que se propaga através de forma circuncêntrica, em meios que tenham massa e elasticidade como sólidos, líquidos ou gasosos, ou seja, não se propaga no vácuo [Fre94]. O som é produzido quando um determinado corpo coloca em movimento uma quantidade (massa) de ar rápido o suficiente para gerar uma seqüência de ondas. É a variação da pressão sobre a massa de ar que causa os diferentes sons, dentre eles os que são combinados para criar a música [Lac96]. As ondas sonoras no ar, causadas por essa variação de pressão, chegam aos ouvidos e fazem o tímpano vibrar. As vibrações são transformadas em impulsos nervosos, levadas até o cérebro e lá decodificadas [Bac97].

Devido à natureza quase-periódica[1] do som de instrumentos musicais melódicos[2], os sons podem ser decompostos em combinações de funções matemáticas primitivas chamadas de função seno ou senóides e cada instrumento produz uma modalidade matemática diferente. O som é

[1]Quase-periódica: que pode ser expressa por uma soma de p funções periódicas.

[2]Instrumentos musicais melódicos: no decorrer da música não trabalha com acordes.

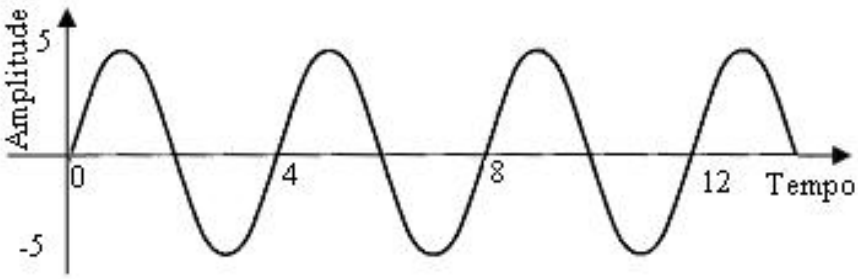


Figura 2.1: Forma de onda sonora.

medido fisicamente por sua intensidade, freqüência e timbre.

2.1.1 Intensidade

A amplitude das oscilações de pressão do ar, chamada de intensidade, é o afastamento da forma de onda a partir da origem, na direção vertical. A potência do som pode ser definida como a percepção dessa intensidade. Quanto maior a amplitude da onda, maior é a quantidade de energia que ela carrega e, consequentemente, maior é o seu volume sonoro [Bac97].

Ao observar a Figura 2.1 é possível identificar a amplitude da onda, no eixo vertical, variando de -5 a 5 unidades no decorrer do tempo.

2.1.2 Freqüência

O número de vezes que as oscilações ocorrem, ou seja, quantos ciclos completos acontecem por unidade de tempo, é chamado de freqüência e a sua percepção é conhecida por tom. Quanto maior for a freqüência de um som, mais agudo ele será. O contrário, quando mais baixa for a freqüência, mais grave o som será. Para descrever a freqüência é utilizada a unidade Hertz (Hz), onde 1Hz corresponde a um ciclo de vibração por segundo. Ao observar a Figura 2.1 é possível ver que o primeiro ciclo termina em 4 unidades, o segundo em 8 e assim por diante [Bac97].

O som pode ser representado por uma soma de diversas ondas individuais chamadas de componentes de Fourier e cada uma corresponde a uma determinada freqüência múltipla da componente inicial. Essas componentes formam uma série conhecida como série harmônica onde o harmônico de ordem zero é chamado de freqüência natural ou fundamental, o segundo

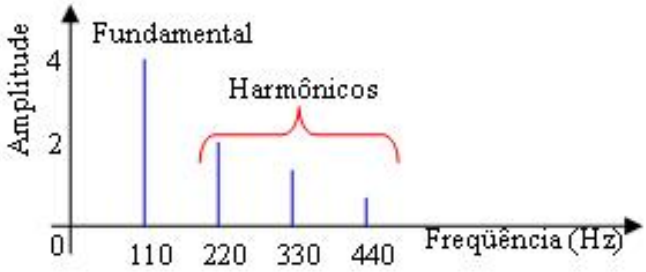


Figura 2.2: Espectro de Freqüência.

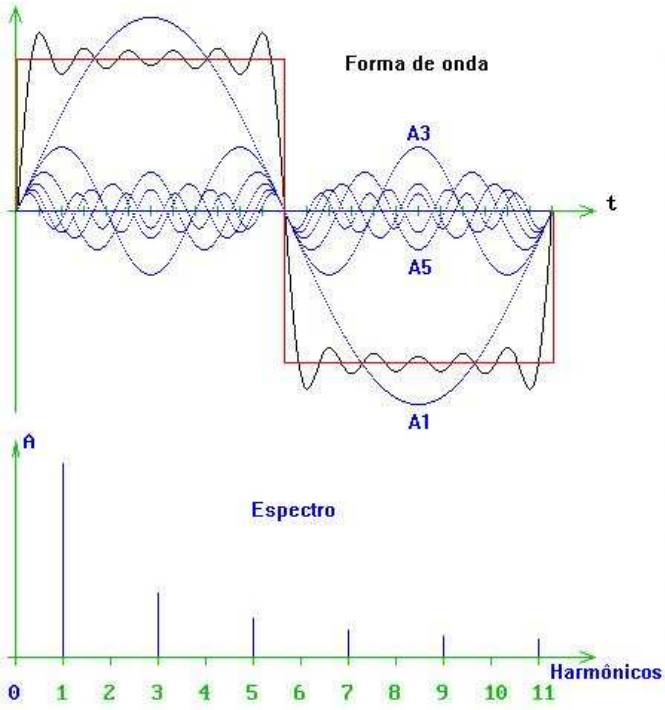
é denominado harmônico de primeira ordem ou primeiro harmônico, o terceiro é chamado segundo harmônico e assim sucessivamente [Lac96].

Diferentes tipos de instrumentos musicais têm número de harmônicos diferentes e específicos de cada instrumento ao soar uma determinada freqüência fundamental. As características espectrais de um instrumento também dependem da intensidade desta freqüência. Um instrumento musical, conhecido por Fagote[3], por exemplo, chega a apresentar apenas dois harmônicos ao soar a freqüência fundamental 523Hz, enquanto que apresenta dez harmônicos quando soa 82Hz [Lac96].

Esses harmônicos são combinados para formar uma onda complexa através de um processo conhecido como síntese de Fourier. O processo inverso, que é determinar os harmônicos que compõem essas ondas complexas, é chamado de análise de Fourier. O exemplo de uma onda complexa, conhecida por onda quadrada, é mostrado em vermelho na Figura 2.3. A representação desta onda, na cor preta, é formada a partir da síntese de Fourier das demais ondas exibidas em azul. Na mesma figura também é mostrado o espectro de freqüência desta síntese de Fourier, ou seja, o espectro de freqüência da onda quadrada.

Em uma amostra de som onde a freqüência fundamental não esteja presente por algum motivo, o valor da diferença absoluta de dois harmônicos adjuntos detectados nessa amostra determina a freqüência fundamental [HV01]. Visualizando a Figura 2.2 é possível confirmar o que foi dito ao observar que a diferença absoluta entre dois harmônicos adjuntos é de 110Hz, correspondendo à freqüência fundamental.

[3]Fagote: instrumento musical de sopro.



onda quadrada

termo	Ampl.	fase
A 0	0	(cc)
A 1	100	0
A 2	0	0
A 3	33	0
A 4	0	0
A 5	20	0
A 6	0	0
A 7	14	0
A 8	0	0
A 9	11	0
A10	0	0
A11	9	0

Figura 2.3: Formação de um sinal complexo.

2.1.3 Timbre

Cada tipo de instrumento musical tem uma espécie de assinatura, um conjunto de características sonoras associado a ele. Embora possam parecer subjetivas, elas têm uma descrição matemática extremamente precisa que torna cada tipo de instrumento único. A mesma freqüência emitida por um violão soa diferente quando produzida por um piano. Isto acontece porque, embora a freqüência fundamental seja a mesma em ambos os instrumentos, o som de instrumentos musicais não é perfeitamente periódico. Portanto, as amplitudes e o tempo de duração de cada um dos harmônicos presentes no som resultante podem variar independentemente. A combinação dessas duas propriedades tem o nome de timbre [DJ97].

A síntese sonora de timbres de instrumentos musicais depende, basicamente, do conjunto de harmônicos utilizados e do envelope sonoro[4], que é formado por transientes[5] de Atack, Decay, Sustain, Release (ataque, decaimento, nível de sustentação e término - ADSR). Cada instrumento musical possui um envelope sonoro característico não só nos transientes de ADSR, como também na altura do som emitido [Dov99].

[4]Envelopes sonoros: designam a variação de intensidade de um som ou nota produzida por um instrumento musical.

[5]Transiente: é a capacidade eletrônica para replicar os rapidíssimos acentos de uma interpretação musical.

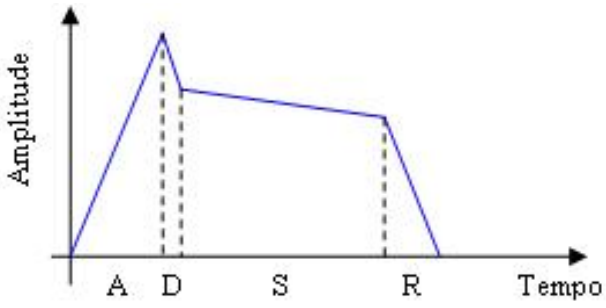


Figura 2.4: Envelope do sinal de audio.

2.2 Distribuição Logarítimica das Faixas de Freqüências

Em média, o ouvido humano é capaz de distinguir cerca de 1400 freqüências discretas, variando entre aproximadamente 20Hz e 20000Hz. Sons fora deste intervalo não são percebidos porque não possuem energia suficiente para excitar o tímpano, ou porque a freqüência é tão alta que o tímpano não consegue perceber [DJ97].

Devido a limitações fisiológicas, o ouvido humano não consegue distinguir entre freqüências com intervalos inferiores ou iguais a uma coma[6], cujo valor é 81/80 (1,0125Hz). Embora exista um número infinito de freqüências, somente é possível definir um número finito de intervalos perceptíveis [Bac97].

2.2.1 Oitavas

A relação entre duas freqüências xHz e yHz, sendo a primeira mais baixa que a segunda, é por definição a razão y/x. Quando dois sons têm relação de freqüência 2:1 esta recebe o nome de oitava. Dentro da faixa de percepção auditiva humana é possível verificar que existem 10 oitavas: 20Hz a 40Hz, 40Hz a 80Hz, 80Hz a 160Hz, 160Hz a 320Hz, 320Hz a 640Hz, 640Hz a 1280Hz, 1280Hz a 2560Hz, 2560Hz a 5120Hz, 5120Hz a 10240Hz e 10240Hz a 20480Hz. Ainda dentro dessa faixa algumas freqüências discretas foram selecionadas para o propósito musical, formando as escalas musicais [Lac96].

2.2.2 Escalas Musicais

Em quantas partes se pode dividir uma oitava ou quantas freqüências podem ter em um intervalo de uma oitava? A escala musical é um conjunto de freqüências musicais e dos intervalos

[6]Coma: nome dado ao valor resultante da relação 81/80.

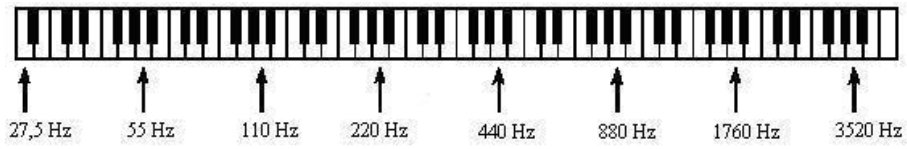


Figura 2.5: Teclado com 7 oitavas e 12 intervalos por oitava.

entre elas, existentes numa oitava. Historicamente, diversas culturas desenvolveram os seus próprios conceitos e regras para a organização dos sons e a escala sempre esteve presente na organização das alturas [Bac97].

Na cultura ocidental, o ponto de partida para o desenvolvimento das escalas e para sua base teórica ocorreu na Grécia Antiga. Pitágoras estudou os modos de vibração de uma corda estendida e descobriu a relação matemática entre os harmônicos. Os primeiros sistemas de afinação foram baseados nos fenômenos relacionados com a série harmônica. Eles baseavam a relação das alturas nas razões dos diversos intervalos obtidos na série harmônica [Moo90].

Assim, qualquer freqüência imaginável no intervalo perceptível para o ouvido humano se aproximaria de alguma das 21 freqüências nesta escala, chamada de justa ou natural, com um erro menor do que uma coma e meia. O intervalo entre duas freqüências sucessivas nunca excede 25/24, que é menor do que três comas e, portanto, perceptível somente para os ouvidos mais treinados [Moo90].

A outra solução, ideal para os instrumentos de teclado, foi dividir a afinação e distribuir este erro, inevitável por causa da forma da divisão da escala, entre freqüências vizinhas. A oitava, inicialmente separada em 21 freqüências na escala natural, foi dividida em apenas 12 intervalos rigorosamente iguais, surgindo a escala igualmente temperada. Nesse sistema, nenhum intervalo, à exceção da oitava, corresponde à afinação natural, mas todos os intervalos resultam suficientemente afinados para serem tolerados pelo ouvido humano [Smi99].

Instrumentos "não temperados" como o violino e o violoncelo são aqueles que não tem uma afinação fixa utilizando a escala natural, enquanto que o violão, guitarra, cavaquinho, piano e teclado, entre outros, são os de afinação fixa ou "temperada". A mais recente música ocidental usa uma escala de 12 intervalos, chamada de escala cromática, onde todos os intervalos são rigorosamente iguais.

2.2.3 Cálculo de Intervalos Igualmente Temperados

Sabendo que a escala igualmente temperada tem 12 intervalos iguais e após o término da oitava a freqüência dobra o seu valor, os intervalos podem ser obtidos pela multiplicação sucessiva de uma determinada freqüência por um fator multiplicador δ (intervalo) até que este seja igual a 2 [Dov99]. O valor de δ é obtido a partir de:

$$f_2 = 2f_1 = \prod_{n=1}^{12} f_1\delta = f_1\delta^{12} \tag{2.1}$$

logo,

$$\delta = \sqrt[12]{\frac{f_2}{f_1}} = \sqrt[12]{2} = 1,0594631 \tag{2.2}$$

Dessa forma, as freqüências que compõem a oitava de uma escala musical igualmente temperada são resultantes de uma progressão geométrica crescente onde o primeiro termo é a freqüência fundamental escolhida e a razão é igual a $2^{1/12}$ [DJ97].

Em 1953 a International Standards Organization (ISO) recomendou a adoção da freqüência 440Hz como a freqüência padrão no mundo e a partir desta é possível obter as outras freqüências da escala musical. Utilizando-se a Equação 2.3 é possível obter qualquer freqüência dentro de um intervalo em uma oitava solicitada, tendo como base a freqüência padrão [Org75].

$$f = 440 \times 2^{m-5}\delta^{(n+2)} = \frac{440 \times 2^m}{2^5} \times \delta^{\left(\frac{n+2}{12}\right)} \tag{2.3}$$

ou

$$f = \frac{440}{32} \times 2^{\left(m+\frac{n+2}{12}\right)} = 13,75 \times 2^{\left(m+\frac{n+2}{12}\right)} \tag{2.4}$$

onde m é o número de oitavas (1-7) e n o número de intervalos (1-12).

Capítulo 3

Teoria Musical

A evolução da formação do sistema musical atualmente utilizado, o sistema temperado, trouxe diversos benefícios como, por exemplo, a possibilidade de transposição perfeita de uma freqüência para qualquer outra [Net06]. Em contrapartida, é correto afirmar que as freqüências no sistema temperado não permitem a criação de intervalos acusticamente perfeitos. Um intervalo temperado de quinta já não tem mais a relação 3:2, embora o erro seja irrelevante para a maioria dos ouvidos humanos, não chegando nem mesmo a proporcionar efeito de batimento[1]. As aproximações feitas pelo temperamento igual, no entanto, certamente trouxeram muito mais benefícios do que o prejuízo que aquele erro possa causar e, por isso, se mantém até os dias de hoje [Lac96].

3.1 Notas Musicais

Uma nota musical é um som cuja freqüência de vibração encontra-se dentro do intervalo perceptível pelo ouvido humano e a música é a combinação, sob as mais diversas formas, de uma seqüência de notas em diferentes intervalos. Entretanto, uma mesma nota emitida por diferentes fontes (ou seja, instrumentos musicais) pode ter a mesma freqüência e ainda assim soar de maneira diferente para quem ouve, como citado anteriormente no item "Timbre" [Med97].

As sete notas musiais existentes, Dó, Ré, Mi, Fá, Sol, Lá e Si, se repetem em intervalos formando oitavas. Uma nota em um intervalo possui o dobro do valor da sua freqüência no intervalo anterior. Dessa maneira, a nota Lá pertencente à quarta oitava com freqüência 440Hz (Lá4), quando dobra o valor da sua freqüência (880Hz) passa a ser a nota Lá5 pertencente à quinta oitava [KD00].

[1]Efeito de batimento: superposição de duas ondas que propagam numa mesma direção em freqüências ligeiramente diferentes

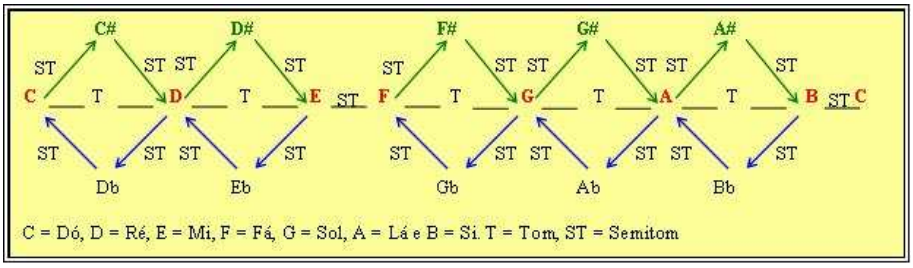


Figura 3.1: Intervalos entre as notas e acidentes musicais.

3.1.1 Tom e Semitom

Os 12 intervalos que compõem a oitava são chamados de semitons. Portanto, existem 12 semitons iguais numa oitava. Dois intervalos (semitons) juntos, formam um tom. Um semitom separa uma nota de um acidente musical ou de outra nota [Lac96].

3.1.2 Acidentes Musicais

Os acidentes musicais, bemol (b) e sustenido (#), alteram o valor da nota em um semitom para baixo e para cima, respectivamente. A Figura 3.1 ilustra uma oitava com seus 12 semitons (notas e acidentes musicais) representados por uma nomenclatura universal conhecida por cifra[2]. Na mesma figura é possível notar que os acidentes C#, D#, F#, G# e A# possuem, respectivamente, a mesma freqüência dos acidentes Db, Eb, Gb, Ab e Bb quando soam na mesma oitava [KD00].

3.2 Padronização das Freqüências Sonoras

De acordo com a equação 2.3 é possível construir uma tabela com as 7 oitavas destinadas ao contexto musical. A Figura 3.2 mostra o valor da freqüência para uma nota ou acidente musical a partir da nota Lá central (A4), que possui a freqüência de 440Hz pertencendo à quarta oitava. As sete oitavas estão representadas pelas colunas e os intervalos pelas linhas [Bac97].

[2]Cifra: escrita simbólica das notas musicais.

NOTAS		1	2	3	4	5	6	7
1	C	32,703196	65,406391	130,81278	261,62557	523,25113	1046,5023	2093,0045
2	C#	34,647829	69,295658	138,59132	277,18263	554,36526	1108,7305	2217,461
3	D	36,708096	73,416192	146,83238	293,66477	587,32954	1174,6591	2349,3181
4	D#	38,890873	77,781746	155,56349	311,12698	622,25397	1244,5079	2489,0159
5	E	41,203445	82,406889	164,81378	329,62756	659,25511	1318,5102	2637,0205
6	F	43,653529	87,307058	174,61412	349,22823	698,45646	1396,9129	2793,8259
7	F#	46,249303	92,498606	184,99721	369,99442	739,98885	1479,9777	2959,9554
8	G	48,999429	97,998859	195,99772	391,99544	783,99087	1567,9817	3135,9635
9	G#	51,913087	103,82617	207,65235	415,3047	830,6094	1661,2188	3322,4376
10	A	55	110	220	**440**	880	1760	3520
11	A#	58,27047	116,54094	233,08188	466,16376	932,32752	1864,655	3729,3101
12	B	61,735413	123,47083	246,94165	493,8833	987,7666	1975,5332	3951,0664

Figura 3.2: Freqüência Sonoras.

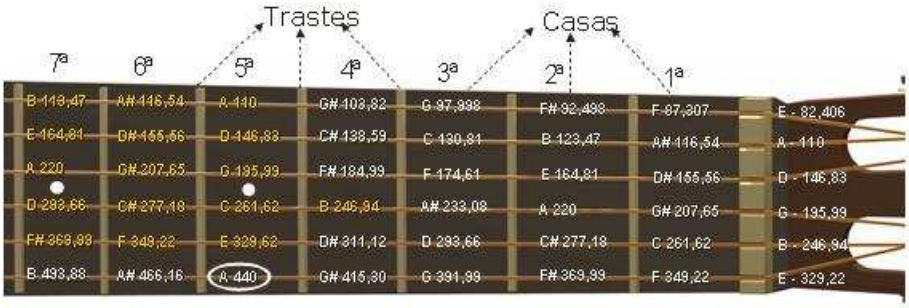


Figura 3.3: Braço de um violão com as notas e freqüências.

3.3 Notas e Freqüências no violão

O braço do violão popular é dividido por trastes formando espaços chamados de casas, como mostrado na Figura 3.3. Dentro dessas casas as notas musicais estão separadas, uma das outras, por um semitom. As seis cordas ficam sobre o braço e são organizadas da mais fina para a mais grossa, ou ainda, da freqüência mais alta para a mais baixa ao visualizar o violão de baixo pra cima.

A afinação deste instrumento é dita "afinação em quartas", porque depois de um intervalo de quatro casas (semitons) a freqüência passa a ser a mesma da corda imediatamente abaixo desta. Assim, quando é pressionada a sexta corda (E = 82,406Hz) na quinta casa, é obtida a mesma freqüência da quinta corda solta (A = 110Hz) [Ols67].

Existe uma variação de 12 casas para cada corda antes de atingir o bojo (a caixa do violão). Como o intervalo entre cada casa é de um semitom e 12 semitons formam uma oitava, ao pressionar uma corda na casa 12 é obtida a mesma nota da corda solta, porém uma oitava acima. A primeira corda do violão (E = 329,22Hz), a mais alta, na casa 12 tem uma freqüência de

659,25Hz, pertencendo a quinta oitava. Assim, é possível concluir que a faixa de freqüência do violão varia entre 82,406Hz (E) a 659,25Hz (E).

3.4 Acordes Musicais

Três ou mais notas tocadas simultaneamente formam um acorde. Os acordes são formados por intervalos harmônicos dando um sentido de harmonia [Alv06]. As formas (desenhos) de acordes são muitas no braço do violão, portanto basta a combinação dos intervalos desejados e digitá-los tomando cuidado para não tornar sua execução impossível. Os acordes maiores e menores são chamados de tríades, pois precisam apenas de 3 notas pertencentes a sua escala.

3.4.1 Acordes Maiores

A partir de uma escala conhecida como escala maior natural serão formados os acordes maiores. Essa escala é construída respeitando a distribuição de Tons e Semitons (T-T-ST-T-T-T-ST). A partir da nota C é possível formar sua escala maior, seguindo a distribuição citada, obtendo a escala C, D, E, F, G, A, B e C. Tomando a primeira, terceira e quinta notas dessa escala, C, E e G, o acorde de Dó maior é formado [Ols67].

3.4.2 Acordes Menores

Da mesma maneira que os acordes maiores, é possível formar os acordes menores a partir de uma escala menor e respeitando a distribuição de Tons e Semitons (T-ST-T-T-ST-T-T). Logo, a escala de Dó menor é C, D, Eb, F, G, Ab, Bb, C. Tomando a primeira, terceira e quinta notas dessa escala, C, Eb e G, o acorde de Dó menor é formado [Ols67]. De uma forma prática, partindo de um acorde maior natural é possível obter um acorde menor diminuindo um semitom da sua segunda nota.

3.4.3 Acordes no Violão

O acorde no violão é representado pelas notas que o formam e a repetição de algumas destas devido à quantidade de cordas que vibram ao mesmo tempo ser maior que as necessárias para a formação dos acordes. Logo, os acordes maiores e menores são formados pela sua tríade e a

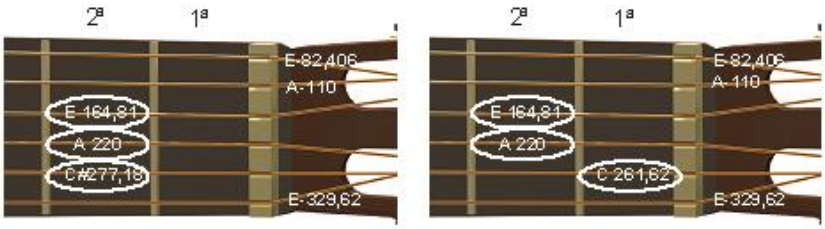


Figura 3.4: Acorde de Lá maior e Lá menor respectivamente.

repetição de algumas notas desta.

A Figura 3.4 ilustra um dos possíveis desenhos no violão para o acorde Lá maior que é composto pelas notas A, C# e E extraídas da sua escala maior A, B, C#, D, E, F#, G# e A, além do acorde Lá menor composto das notas A, C e E extraídas da sua escala menor A, B, C, D, E, F, G e A, respectivamente.

Observando a Figura 3.4 é possível perceber a repetição de duas notas "E" e uma nota "A", além de verificar o decréscimo de um semitom na segunda nota do acorde maior (C#) para a formação do acorde menor.

Capítulo 4

Sistema de Reconhecimento

O Sistema de Reconhecimento desenvolvido neste trabalho é capaz de processar arquivos com amostras de áudio, contendo apenas uma nota musical por arquivo, gravados, previamente, a partir de um violão elé trico. As notas se encontram no domínio do tempo e para fazer a aná lise do sinal de áudio é foi necessário transformá-las para o domínio da freqüência, utilizando a transformada rápida de Fourier [Bor98]. A partir do sinal obtido no domínio espectral, é possível analisar, entender e diferenciar as amostras de áudio para poder desenvolver um algoritmo capaz de identificar os padrões referentes a informações musicais [OJ02]. No passo seguinte classificam-se as notas musicais de acordo com o resultado apresentado pelo algoritmo do passo anterior.

A Figura 4.1 representa graficamente as etapas para que um sinal de áudio analógico seja adquirido e identificado pelo Sistema de Reconhecimento.

Com base no aprendizado dessa experiência, os mesmos passos são aplicados, porém ajustados, para o reconhecimento de acordes musicais. Novas amostras são gravadas contendo, desta vez, um acorde por arquivo. Mais uma vez, a transformada rápida de Fourier é empregada para converter as amostras para o domínio da freqüência. Além disso, é necessário elaborar mais outro algoritmo capaz de identificar padrões referentes a informações musicais, desta vez para acordes, e um outro para classificá-los a partir do resultado encontrado pelo algoritmo anterior.

4.1 Gravação das Amostras de Áudio

Antes que as gravações das amostras fossem realizadas, o violão utilizado na implementação do projeto foi submetido a uma afinação minuciosa de suas cordas, com a ajuda de um afinador eletrônico, para validar o sistema proposto. Para a captura do som foi estabelecida uma

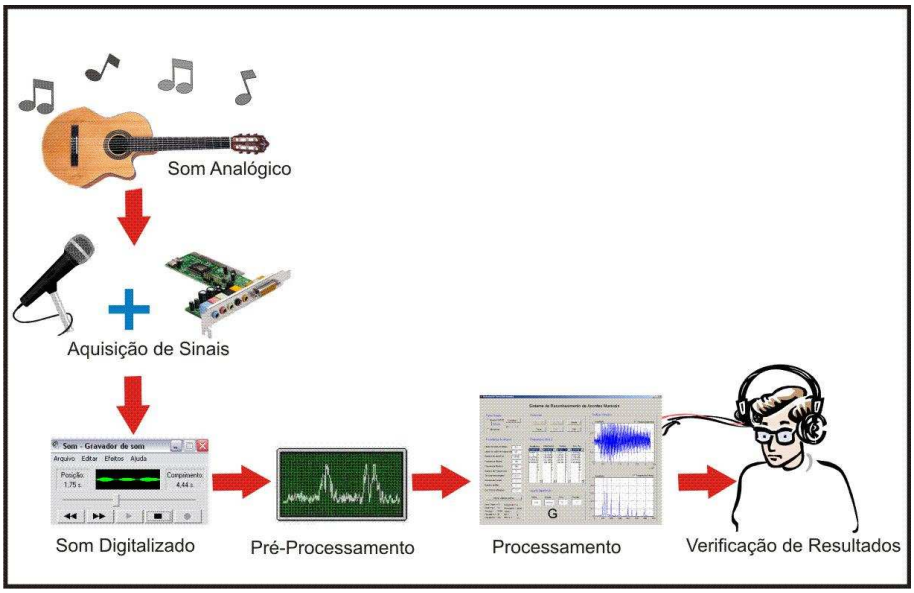


Figura 4.1: Sistema de Reconhecimento de Acordes.

conexão, através de um cabo estéreo, da saída de áudio do violão elétrico com a entrada de microfone da placa de áudio do computador. Tal procedimento foi necessário para evitar que ruídos no ambiente pudessem distorcer o sinal que seria gravado [Wat94]. Foi utilizado o aplicativo "Gravador de Som", presente na instalação padrão do sistema operacional Windows, para realizar as gravações das amostras de áudio em arquivos. Esse software armazena os sinais de áudio em formato do tipo wave com 44.100Hz, 16 bits e em dois canais.

Para poder conseguir analisar todas as possibilidades de notas executáveis do violão, foram armazenados os sinais de áudio da corda mais grave (E – 82,40Hz) vibrando solta, depois pressionada na sua segunda casa e assim por diante, até alcançar a corda mais aguda vibrando pressionada na sua 12ª casa (E - 659,25Hz). Com isso foi possível obter um conjunto de 37 arquivos contendo sinais de notas musicais amostrados de um violão tocados com a mão.

De forma semelhante foram armazenados os principais formatos de acordes executáveis no violão, sendo que estes podem ser maiores, menores e sustenidos. A base de acordes possui 49 arquivos com sinais de áudio também tocados em um violão com a mão. Dessa maneira, o total de arquivos gerados para análise contendo sinais de áudio que abrangem notas e acordes musicais é de 86.

4.2 Leitura das Amostras de Áudio

O Sistema de Reconhecimento faz a leitura das amostras de áudio levando em consideração que estas foram gravadas com freqüência de amostragem de 44.100Hz, número de bits igual a 16 e em dois canais. O arquivo é lido na íntegra e armazenado em uma variável do tipo vetor, na memória, onde sofre alguns ajustes: primeiro são removidas as amplitudes consideradas como ruído no início do sinal conforme os parâmetros informados na aplicação pelo usuário. Caso não seja definido o limiar de ruído no tempo, este, por padrão, recebe o valor 0,2 de amplitude (menor valor, entre os limites normalizados 0 e 1, que elimina o ruído no tempo); em seguida, é considerado o número de amostras também informado na aplicação, que, por padrão, processa 32.768 amostras (número de amostras que, de acordo com os testes realizados, permite melhor análise do sinal); por fim, é feita a normalização deste sinal.

Além dos cálculos efetuados, o Sistema também exibe o gráfico da amostra, já processada, em função do tempo para uma eventual análise visual e permite a sua audição para confirmar que a amostra processada é a mesma que foi gravada.

4.3 Ferramenta Aplicada ao Problema

Inicialmente foi necessário escolher uma das ferramentas na área de processamento digital de sinais como, por exemplo, Recursive Discrete Fourier Transform – RDFT, Recursive Least Squares – RLS e Fast Fourier Transform – FFT, para a transformação dos sinais de domínio do tempo para o domínio da freqüência [Mar03]. A ferramenta que se mostrou mais eficiente (menor custo computacional) para a estimativa de amplitude de um sinal composto de harmônicos de uma freqüência fundamental foi a FFT devido a sua rápida velocidade na execução desta tarefa, o que era imprescindível para alcançar o objetivo de se obter respostas em tempo real.

4.3.1 Transformada Rápida de Fourier

Um sinal periódico qualquer $x_p(t)$ com período $T > 0$ para $\forall\, t$, ou seja, $x_p(t) = x_p(t \pm T)$ pode ser decomposto em componentes senoidais a partir da Série Trigonométrica de Fourier, isto é,

$$x_p(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left[a_k \cos\left(k\omega_0 t\right) + b_k \mathrm{sen}\left(k\omega_0 t\right)\right] \tag{4.1}$$

sendo

$$\omega_0 = \frac{2\pi}{T} \tag{4.2}$$

a freqüência fundamental e T o período [OS89]. As constantes, ou coeficientes de Fourier, a_k e b_k são obtidas através de:

$$a_k = \frac{2}{T}\int_0^T x_p(t)\cos(k\omega_0 t)dt \tag{4.3}$$

e

$$b_k = \frac{2}{T}\int_0^T x_p(t)sen(k\omega_0 t)dt \tag{4.4}$$

A equação (4.1) pode ser reescrita na forma exponencial, ou seja,

$$x_p(t) = \sum_{k=-\infty}^{\infty} a_k e^{jk\omega_0 t} \tag{4.5}$$

sendo

$$a_k = \frac{1}{T}\int_0^T x_p(t)e^{-jk\omega_0 t}dt \tag{4.6}$$

os coeficiente complexos da Série de Fourier que multiplicam cada termo da expansão. O coeficiente a_0 é o valor DC ou componente contínua de $x_p(t)$ que representa o valor médio de $x_p(t)$ em um período, seu valor é fornecido por:

$$a_0 = \frac{1}{T}\int_0^T x_p(t)dt \tag{4.7}$$

A Sérier de Fourier é aplicada à análise de sinais periódicos no tempo. A série de um sinal periódico possibilita a análise da composição harmônica do mesmo. Para sinais aperiódicos a resposta em freqüência é obtida através da Transformada de Fourier [HV01]. Sabendo que em sinais aperiódicos $x(t) = 0$ para $\forall\, t \geq \left|\frac{T}{2}\right|$ e considerando $a_k T = X(k\omega_0)$, com $k\omega_0 = \omega$, tem-se:

$$X(\omega) = \Im\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{4.8}$$

sendo $x(t)$ um sinal do tempo, contínuo e aperiódico. O resultado da transformada, $X(\omega)$, é um sinal no domínio da freqüência, contínuo e aperiódico.

Para sinais periódicos discretos $x_p[n]$, isto é, $x_p(n) = x_p(n \pm N)$ com período N, utiliza-se a Série de Fourier discreta,

$$x_p[n] = \sum_{k=-\infty}^{\infty} a_k e^{jk\Omega_0 n} \tag{4.9}$$

sendo

$$\Omega_0 = \frac{2\pi}{N} \tag{4.10}$$

a freqüência fundamental e N o período. Os coeficientes a_k, para todo e qualquer k (com $k = 0, 1, 2...N-1$), podem ser considerados como uma seqüência periódica, com período N, da qual é possível extrair apenas N termos para se obter a representação completa por Série de Fourier de $x[n]$. Portanto, para sinais discretos, a equação (4.1) e os coeficientes a_k podem ser escritos respectivamente como:

$$x_p[n] = \sum_{k=0}^{N-1} a_k e^{jk\Omega_0 n} \tag{4.11}$$

e

$$a_k = \frac{1}{N}\sum_{n=0}^{N-1} x_p[n]e^{-jk\Omega n} \tag{4.12}$$

A Sérier de Fourier discreta é aplicada à análise de sinais periódicos discretos no tempo. Para sinais aperiódicos discretos no tempo a resposta em freqüência é obtida através da Transformada Discreta de Fourier - DFT. Sabendo que em sinais aperiódicos $x[n] = 0$ para $\forall\ |n| \leq N$ e considerando que $X(k) = a_k$, conclui-se que a DFT é obtida pela mesma expressão que fornece os coeficiente da Série Discreta de Fourier, ou seja,

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\, e^{-jk\Omega n} = \frac{1}{N}\sum_{n=0}^{N-1} x[n]\, e^{-\frac{j2\pi kn}{N}} \tag{4.13}$$

uma função discreta e aperiódica, enquanto que $X[k]$ é uma função complexa, discreta e periódica, com Ω variando entre 0 e 2π. Fazendo $W_N = e^{-\frac{j2\pi}{N}}$, tem-se:

$$DFT\left\{x[n]\right\} = X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]W_N^{-kn} \tag{4.14}$$

onde W_N^{kn} é denominado fator de giro da DFT. Os fatores de giro definem pontos sob o círculo unitário no plano complexo z e sã o igualmente distribuídos em torno do círculo e incrementados de uma freqüência definida por Fs/N, em que Fs é a freqü ência (ou taxa) de amostragem e N o número de amostras.

A Transformada Rápida de Fourier – FFT é um método computacional (algoritmo) que calcula, de forma extremamente rápida, a DFT. Para comparar, suponha um número de amostras igual a 1.024, ou seja, $N = 2^{10}$. A DFT consome tempo de CPU (processamento de computador) na ordem $O(N^2)$ o que resulta em 1.048.576 operações, enquanto a FFT tem a ordem $O(N \log_2^N)$ resultando em 10.240 operações. Em segundos, com $N = 10^6$, se a FFT, utilizando o processador hipotético X gasta 30 segundos, a DFT gastaria duas semanas no mesmo processador [Smi99].

O software utilizado para a implementação do projeto, o MatLab, possui uma função denominada de FFT que realiza a transformada rápida de Fourier a partir de um sinal armazenado em um vetor. Assim, foi possível transformar os sinais de áudio amostrados no tempo para

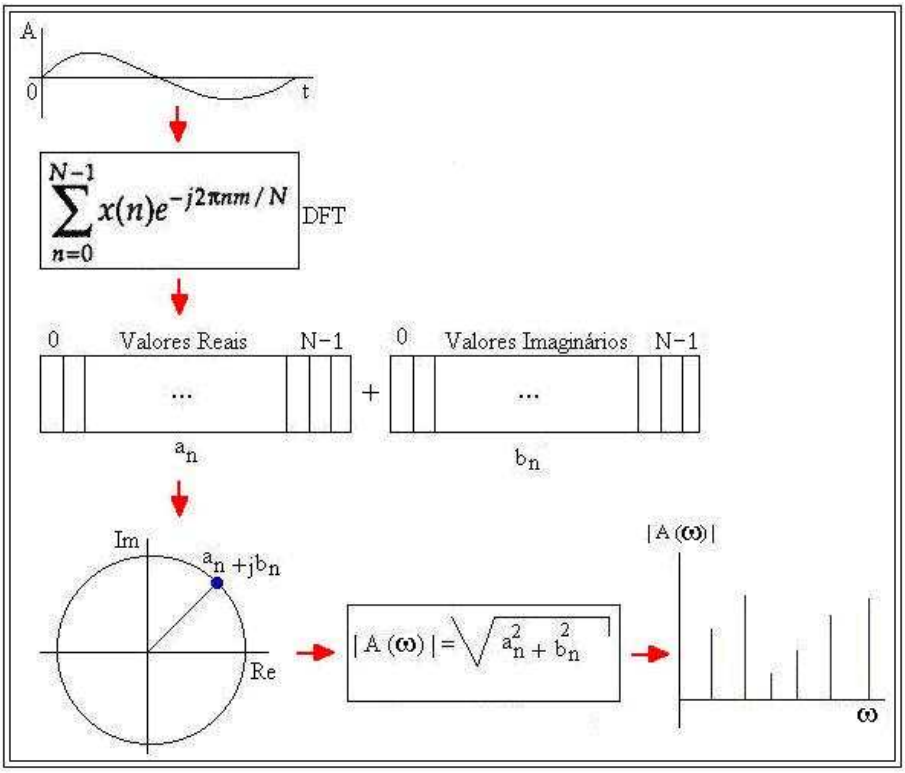


Figura 4.2: Processo para obtenção da Transformada Discreta de Fourier.

o domínio da freqüência, abstraindo os detalhes de implementação do algoritmo, que mostrou um bom desempenho na realização desta tarefa.

Na Figura 4.2, a representação do sinal de uma amostra de áudio que está no domínio do tempo é submetida à expressão da transformada discreta de Fourier, gerando como saída um vetor com valores complexos. A representação deste vetor é exibida na mesma figura em uma circunferência pertencente ao plano complexo Z, onde o eixo das abscissas representa os valores reais e o das ordenadas os valores imaginários do vetor complexo. As amplitudes das componentes espectrais do sinal são obtidas a partir do módulo, ou valor absoluto, deste vetor complexo.

4.3.2 Parâmetros de Análise Aplicados na Resposta da FFT

Para tornar ainda melhor o desempenho do Sistema de Reconhecimento, foram incluídos e ajustados alguns parâmetros na análise do resultado da transformada:

- as freqüências mínima e máxima definidas, como 80Hz e 700Hz respectivamente, e podem ser alteradas a qualquer momento na interface do Sistema. Tal recurso, proporciona a exclusão

de possíveis sub-harmônicos[1] no sinal, espelhamentos de freqüências e diminuição do espectro útil;

- uma normalização das amplitudes é efetuada, tornando possível a comparação dos espectros de freqüência na mesma escala;

- por fim, o limiar de ruído na freqüência que elimina possíveis distorções e permite que apenas as freqüências que possuam amplitude maior que ele sejam analisadas. Como padrão o seu valor é 0,1 de amplitude (menor valor, entre os limites normalizados 0 e 1, que elimina o ruído no tempo), podendo também ser alterado a qualquer instante na interface do Sistema.

Assim como no tópico da leitura das amostras, além dos cálculos efetuados anteriormente, o Sistema exibe um gráfico das amplitudes em função da freqüência, depois que a resposta da transformada de Fourier é submetida à aplicação dos parâmetros citados.

4.4 Detecção dos Picos das Freqüências

A partir da resposta da transformada de Fourier já filtrada e ajustada, é necessária a obtenção de todos os picos de freqüências válidos deste vetor de amplitudes. Tais picos representam uma freqüência fundamental e seus harmônicos, caso o sinal seja uma amostra de nota musical ou seis freqüências fundamentais e seus respectivos harmônicos se tratando da amostra de um acorde.

Para atingir o resultado esperado, foi elaborado um Sistema Especialista que percorre todo o vetor de amplitudes em busca do maior valor (pico) em cada região no espectro. Inicialmente, o primeiro valor do vetor de amplitudes é considerado um pico candidato. Caso o elemento seguinte possua um valor de amplitude maior que o atual, o mesmo ocupará a vaga de pico candidato, substituindo o anterior. Esse processo se repete até ser encontrado um valor de amplitude menor que o pico candidato atual.

Como os sinais tratados nesse projeto são amostras de áudio musical, o valor do multiplicador (δ_i = 1,0594631) é utilizado para obter os intervalos na escala cromática natural temperada e a tabela das freqüências sonoras, conforme mostrados na Equação 2.3 e Tabela 3.1 respectivamente.

[1]Sub-harmônico: harmônico cujo valor é a metade do valor da freqüência inicial ao invés do dobro. Deste modo os sub-harmônicos são harmônicos com freqüências mais baixas que as fundamentais.

Assim, o pico candidato é eleito, somente quando a diferença entre a sua freqüência e a última eleita for maior que a multiplicação do valor de ä$_i$ pela última freqüência eleita subtraído dela mesma, ou seja:

$$f_p - f_e > f_e\delta_i - f_e = f_e\left(\delta_i - 1\right) \tag{4.15}$$

ou ainda

$$f_p > f_e + f_e\left(\delta_i - 1\right) = \delta_i f_e \tag{4.16}$$

sendo f_p a freqüência do pico candidato e f_e a última freqüência eleita.

Portanto, o pico candidato será eleito caso a sua amplitude seja a maior depois de um intervalo logarítmico formado entre a última freqüência eleita e a freqüência candidata. Por exemplo: suponha que a última freqüência é 440Hz e a freqüência do pico candidato seja 450Hz. Então, o resultado booleano da operação relacional {[450-440] > [(440 x 1,0594631) – 440]} é (10 > 26,16), que é falso. Logo, o pico candidato não é eleito e o Sistema seguirá a procura de outro pico para ser o candidato. Este resultado pode ser verificado visualmente na Tabela 3.1, onde a próxima freqüência válida depois de 440Hz é 466,16Hz.

Ainda no algoritmo de detecção de pico das freqüências é possível definir quantos picos serão considerados. Por uma questão lógica, a quantidade de picos padrão definida pelo Sistema é de seis freqüências, pois é o número máximo de cordas vibrando ao mesmo tempo no violão. Porém, esse parâmetro pode ser modificado pela interface do Sistema a qualquer momento. Esses picos detectados são exibidos na interface do Sistema.

4.5 Classificação dos Picos das Freqüências

Para descobrir como classificar os picos das freqüências referentes aos acordes musicais, foi feita a classificação das notas, pois estas apresentam apenas uma freqüência fundamental.

4.5.1 Classificação das Notas Musicais

Observando-se os espectros de freqüências das amostras de notas percebe-se que nem sempre a primeira freqüência é a fundamental e que esta pode até não aparecer no espectro. Apesar de tais fatos ocorrerem, foi possível fazer a classificação, pois as freqüências restantes são múltiplas da fundamental.

Assim, o Sistema de Reconhecimento de notas supõe que a primeira freqüência é a fundamental, calcula a diferença entre dois harmônicos consecutivos e se o valor da suposta freqüência fundamental for igual a esta diferença, a suposição passa a valer como verdade. Caso os resultados sejam diferentes, será necessário calcular a diferença entre outros dois harmônicos consecutivos, e se estas diferenças tiverem o mesmo valor, esta será a freqüência fundamental.

Os valores das freqüências da primeira oitava foram previamente armazenados em um vetor na memória, onde a notação de cada uma das notas musicais está associada a cada posição. A freqüência fundamental é repetidamente dividida por dois até obter o mesmo valor de uma destas doze freqüências armazenadas no vetor. A posição do vetor correspondente a esta comparação indica a nota executada. Este algoritmo foi aproveitado posteriormente no módulo de afinação do instrumento.

4.5.2 Classificação Dos Acordes Musicais

Na classificação dos acordes foram armazenadas, previamente e de forma ordenada, as 49 possibilidades de combinações de acordes maiores, menores e sustenidos em uma matriz. Esta possui seis colunas, cada uma correspondendo à freqüência que cada corda pode emitir considerando os acordes tratados neste trabalho, e quarenta e nove linhas onde cada uma delas representa um acorde. Então, a primeira coluna representa a sexta corda, a segunda coluna a quinta corda até chegar a primeira corda que é a sexta coluna.

O Sistema analisa as freqüências iniciando da mais baixa (sexta corda) para a mais alta (primeira corda), pois assim foram armazenados os picos no processo de detecção e da mesma forma estão na matriz de acordes. É feita uma busca seqüencial da primeira freqüência contida no vetor de picos na primeira coluna da matriz. Desta maneira é identificado um conjunto de linhas (acordes) prováveis. Somente levando em consideração este conjunto de linhas da matriz é realizada a busca do segundo pico de freqüência na segunda coluna. Esse processo se repete até que reste apenas uma linha que corresponda a todos os picos detectados. A posição desta linha na matriz indica o acorde executado.

Além da busca realizada, o Sistema exibe as freqüências padronizadas da linha que correspondeu ao vetor de picos detectados, mostra o percentual de erro para cada freqüência, além de mostrar o acorde reconhecido na notação musical cifra e informar o formato que o mesmo foi executado.

4.6 Execução em Tempo Real

Após ter obtido sucesso no Sistema com o reconhecimento das amostras de áudio armazenadas em arquivo, foi possível submetê-lo à execução em tempo real, ou seja, tocar um acorde no violão e o Sistema imediatamente informar qual acorde foi reconhecido.

Exceto o processo de leitura das amostras em arquivo, todos os módulos que foram utilizados para o reconhecimento nas amostras armazenadas são utilizados em tempo real e o módulo que antes era utilizado para reconhecimento de notas passou a ser utilizado como afinador.

Para obter um bom resultado com o Sistema de Reconhecimento é preciso realizar a afinação das cordas do violão a ser utilizado com o afinador disponível no Sistema. Isso ocorre devido a freqüência de referência, tomada como base nos afinadores eletrônicos comerciais, poder variar. Um exemplo disso é uma marca X de afinador tomar por base a freqüência 415Hz e uma outra Y usar 455Hz para a nota Lá 440Hz.

O armazenamento das amostras de áudio é feita de um em um segundo devido a uma limitação imposta pelo software onde foi realizada a implementação. Este fato resulta num pequeno atraso da exibição do resultado final, indesejado inicialmente neste projeto mas intransponível devido ao software utilizado.

Além dos parâmetros para análise já disponíveis quando foram reconhecidos arquivos de áudio, são acrescentados mais outros três parâmetros que podem variar de acordo com o usuário: a taxa de amostragem, o número de canais e de bits, que, respectivamente, possuem valores 44.100Hz, 2 e 16 por padrão.

Os demais resultados, exibições e conclusões são os mesmos anteriormente informados quando o Sistema de Reconhecimento trabalhava com arquivos de áudio contendo amostras de notas e acordes musicais.

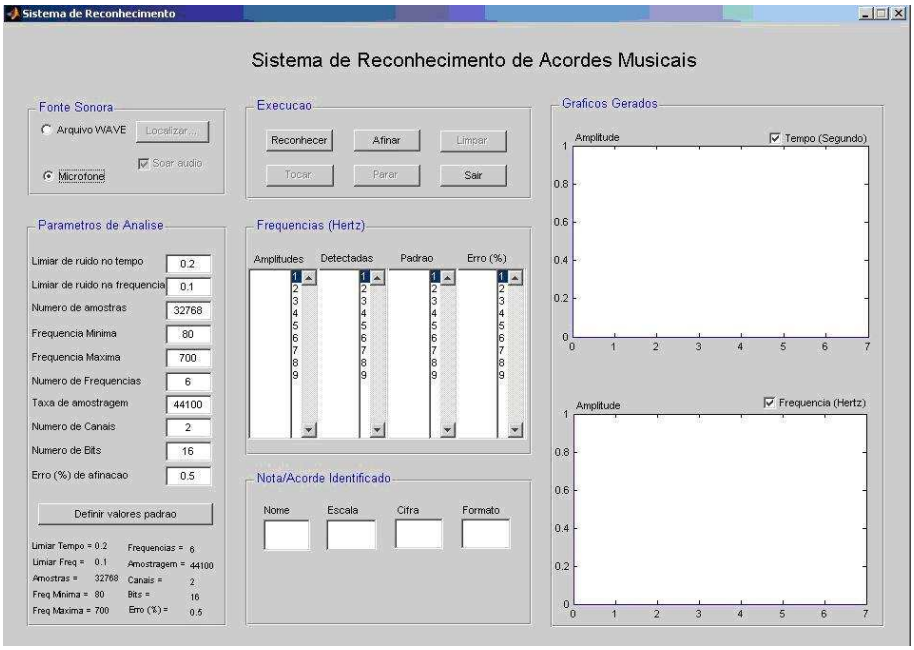


Figura 4.3: Tela do Sistema de Reconhecimento.

4.7 Como Executar o Programa

Para execução do Sistema é necessário ter instalado o software Matlab no computador, pois a compilação dos arquivos é feita por ele. Dentro do Matlab, aponte para o diretório que contém os arquivos onde estão escritos o Sistema de Reconhecimento. Na janela de comandos digite: siram <enter>. A Figura 4.3 será exibida.

Apesar do Sistema ter sido construído com o objetivo de ser interativo e auto-explicativo, o funcionamento pode ser explicado de forma resumida da seguinte forma: na direita são mostrados gráficos gerados no tempo (parte superior) e/ou na freqüência (parte inferior). É possível escolher exibir ou não estes gráficos; ao centro, na parte inferior, são mostrados os resultados dos picos detectados, as freqüências padrão para uma determinada ocorrência, a taxa de erro entre o que foi detectado e o padrão, o acorde reconhecido com seu nome, escala que pertence, o seu formato e a sua cifra. Ainda ao centro, porém na parte superior, existem os botões de execução do software, onde é possível (de acordo com a fonte sonora escolhida) executar o reconhecimento, a afinação, parar o reconhecimento ou afinação, limpar a tela, tocar o que já foi reconhecido e sair do aplicativo; no lado esquerdo inferior é possível editar todos os parâmetros de análise (também de acordo com a fonte sonora escolhida) que foram explicados anteriormente. Por fim, na esquerda superior é possível escolher e/ou localizar a fonte sonora que será submetida ao reconhecimento.

A Figura 4.4 mostra o Sistema depois de ter reconhecido um arquivo contendo uma amostra

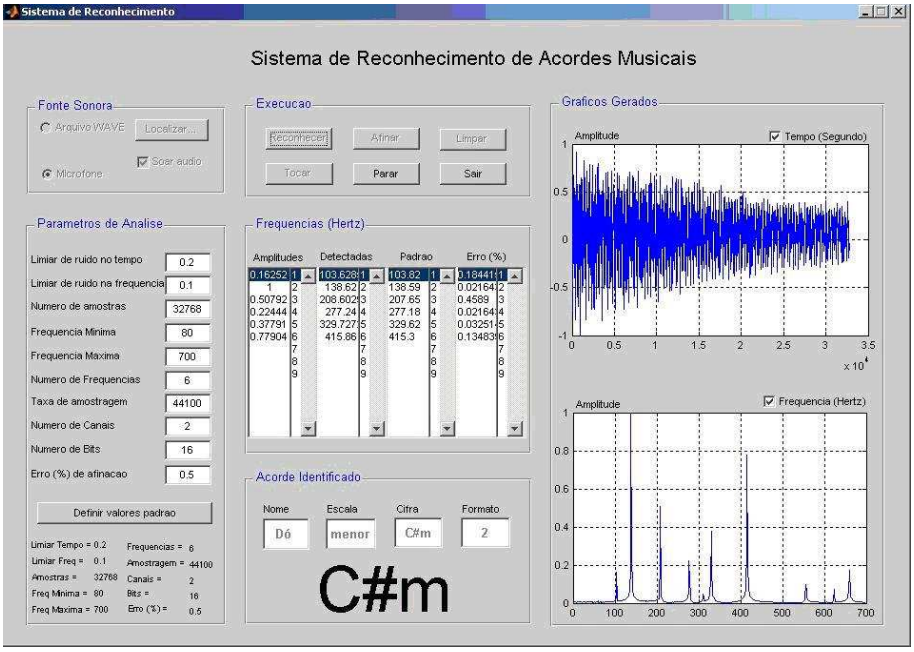


Figura 4.4: Sistema reconhecendo o acorde Dó sustenido menor.

de áudio musical referente ao acorde Dó sustenido menor no seu segundo formato.

4.8 Especificação de Equipamentos e Softwares Utilizados

A implementação da solução do Sistema de Reconhecimento de acordes utilizou os recursos de hardware e software descritos a seguir:

1. Notebook Acer TravelMate 4652LMI:
   - 80GB de HDD;
   - 512MB de RAM DDR2 400MHz dual channel;
   - Processador Intel Pentium M 740 2MB L2 cache, 1.73GHz, 533MHz FSB;
   - Placa de áudio Realtek AC'97.
2. Afinador Eletrônico Cort E205;
3. Violão D'Giorgio estudante n° 18;
4. Sistema Operacional Microsoft Windows XP Professional - Versão 2002, Service Pack 2;

5. MatLab, the language of technical computing - Versão 7.0.0.19920 (R14) de 06 de maio de 2004.

Capítulo 5

Testes e Resultados

5.1 Definição dos Parâmetros

Depois da conclusão do Sistema partiu-se para os testes e aferição de resultados. Os primeiros testes foram realizados variando os parâmetros de entrada para encontrar um conjunto de valores mais adequado ao Sistema de Reconhecimento.

Os parâmetros foram testados de acordo com cada Etapa do processo de reconhecimento. Na Etapa de Aquisição do Sinal, foram testados os parâmetros taxa de amostragem, número de canais e número de bits; na Etapa de Análise da Amostra no Tempo foram avaliados os parâmetros limiar de ruído no tempo e número de amostras; e na Etapa de Análise da Amostra na Freqüência, os parâmetros limiar de ruídos na freqüência, freqüência mínima e freqüência máxima. Por fim, foram definidos os parâmetros número de freqüência, que interfere apenas na visualização dos resultados das freqüências detectadas, e erro de afinação, que define a porcentagem de erro aceitável na afinação das cordas.

O número de bits influencia na qualidade da amostra de áudio, pois é utilizado no processo de quantização[1] do sinal. Dentre os valores possíveis e testados, 8, 16 e 24, o valor 8 foi suficiente para relacionar qualidade e velocidade na execução dos testes, pois não prejudicou a qualidade do som e foi mais rápido do que os demais valores. Para o número de canais foi definido o valor 1, pois o violão não emite som estéreo. A taxa de amostragem influencia na resolução espectral (quando relacionado ao número de amostras) e no número de freqüências possíveis de serem detectadas (teorema de Nyquist)[2]. Dentre os valores possíveis e testados, 8000, 11025,

[1]Quantização: em processamento de sinais, quantização é o processo de atribuição de valores discretos para um sinal cuja amplitude varia entre infinitos valores.

[2]Teorema de Nyquist: a taxa de amostragem deve ser igual ou maior do que duas vezes a maior freqüência do sinal que está sendo digitalizado.

22050 e 44100, o valor 8000 atendeu ao resultado esperado, pois consegue capturar todas as freqüências emitidas pelo violão e, quando relacionado ao número de amostras adotado (8000), cria um intervalo de freqüências aceitável (1Hz).

Foi estabelecido o valor 8000 para o número de amostras, pois este consegue capturar todas as notas existentes no acorde durante o período de sustentação na sua execução. Tal valor foi usado por obter a melhor resolução espectral para a taxa de amostragem igual a 8000 e devido ao MatLab não permitir que a quantidade de amostras analisadas seja maior que a taxa de amostragem. O limiar de ruído no tempo que melhor eliminou os ruídos sem prejudicar a detecção das notas foi 0,1. Valores abaixo do adotado tornam o Sistema muito sensível a ruído ambiente e, o contrário, inibe a detecção de algumas freqüências por interpretá-las como ruído.

A análise aplicada ao limiar de ruído no tempo vale para o limiar de ruído na freqüência, cujo valor diante dos testes realizados foi estabelecido em 0,2. Os parâmetros freqüência máxima e freqüência mínima se aplicam como um filtro passa faixa onde a freqüência inicial é a mínima, 80Hz, e a final é a máxima, 700Hz. Estes limites foram adotados porque o violão soa freqüências fundamentais dentro deste intervalo.

Todos estes parâmetros foram definidos como padrão para a execução do Sistema aplicado ao violão. Porém, os mesmo podem ser alterados para possibilitar análises e testes em outras situações.

5.2 Módulo de Afinação

Para avaliação dos resultados do Módulo de Afinação, todas as seis cordas do violão foram desafinadas aleatoriamente e, em seguida, afinadas com o Sistema. O resultado desta operação foi comparado com o resultado usando o afinador eletrônico utilizado neste trabalho e, ambos foram idênticos.

O espectro de freqüências da Figura 5.1 mostra a corda Lá desafinada com o valor da freqüência fundamental detectada pelo Sistema igual a 107,66Hz. Este valor e os valores dos demais harmônicos detectados não correspondem aos valores padrão da nota Lá, cuja freqüência fundamental é 110Hz. Portanto, o Sistema orienta que a ação a ser realizada pelo executor é apertar a corda para alcança a nota exibida, neste caso a nota Lá.

Seguindo a orientação do Sistema, a corda Lá foi apertada até que a mesma alcançasse o

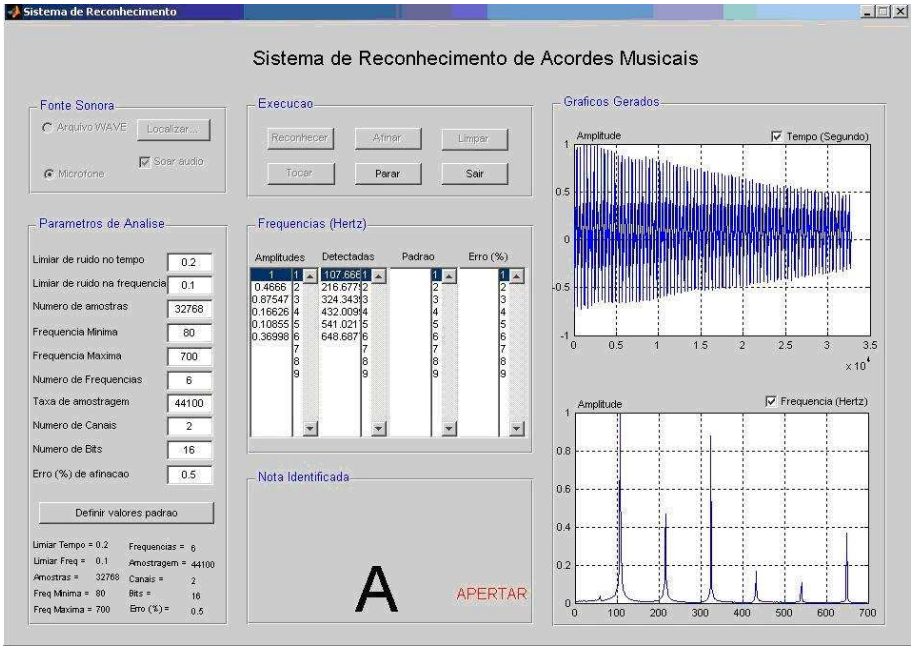


Figura 5.1: Corda Lá desafinada.

seu valor de freqüência padrão e fosse reconhecida. Observando a Figura 5.2, os valores da freqüência fundamental e dos harmônicos correspondem aos valores padrão das freqüências da corda Lá. Por ter tido, nesta execução, um valor de amplitude menor que o valor do limiar de ruído na freqüência atribuído nos parâmetros de análise, a freqüência 550Hz (que correspondente ao quarto harmônico da corda Lá 110Hz) não foi identificada pelo Sistema.

As demais cordas foram submetidas ao mesmo teste e o processo de afinação mostrou-se satisfatório em todos os casos. Assim, o módulo de afinação do Sistema de Reconhecimento obteve 100% de sucesso na afinação de cordas para violão elétrico.

5.3 Módulo de Reconhecimento dos Acordes

Um dos principais problemas no reconhecimento de acorde é a digitação correta de todas as notas envolvidas no mesmo. Quando os testes são realizados a partir de amostras de áudio previamente armazenadas em arquivos, vários parâmetros de análise, como início e fim do sinal, já estão embutidos nesta amostra, além de ser possível armazenar apenas as mais convenientes.

Os 49 acordes armazenados em arquivos foram submetidos a testes no Sistema e o reconhecimento foi realizado com sucesso em 100% deles. A Figura 5.3 exibe o Sistema reconhecendo o acorde Lá menor a partir da amostra de áudio armazenada no arquivo do tipo wave correspon-

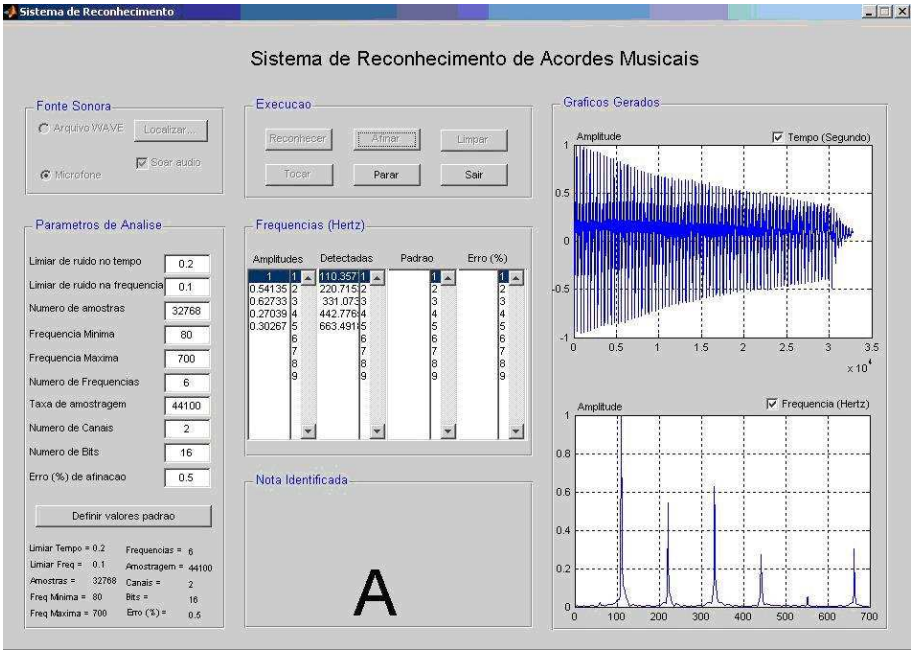


Figura 5.2: Corda Lá afinada.

dente. Nela é possível visualizar uma onda com comportamento regular no gráfico da amostra no tempo implicando em picos bem definidos no espectro de freqüências.

O sucesso no reconhecimento do acorde em tempo real depende da forma de digitação executada, de qual parte do envelope ADSR o acorde é capturado e a intensidade com que é tocado. Variações nestes itens podem fazer o Sistema gerar diferentes gráficos da forma de onda e de espectro de freqüência, além da possibilidade de não identificação do acorde tocado. Esta variação pode ser percebida ao comparar as Figuras 5.3 e 5.4.

Para reconhecer o acorde Ré maior no formato 1 foi necessário adaptar o Sistema à forma habitual que os músicos executam este acorde neste formato, tocando todas as cordas ao mesmo tempo. Neste contexto, a sexta corda (Mi) é tocada, porém esta nota não pertence ao acorde Ré maior. Portanto, para que o Sistema realizasse o reconhecimento deste acorde, esta nota foi adicionada ao conjunto de notas do acorde Ré maior no formato 1, fazendo parte dele neste Sistema. A nota Mi, ao ser tocada, acrescentou novas freqüências ao acorde Ré Maior, além da sua fundamental.

Observando a Figura 5.5, a freqüência fundamental da nota Mi, 82,0953Hz, e seu primeiro harmônico, 164,190Hz, foram detectados na execução do acorde Ré maior, formato 1, conforme esperado.

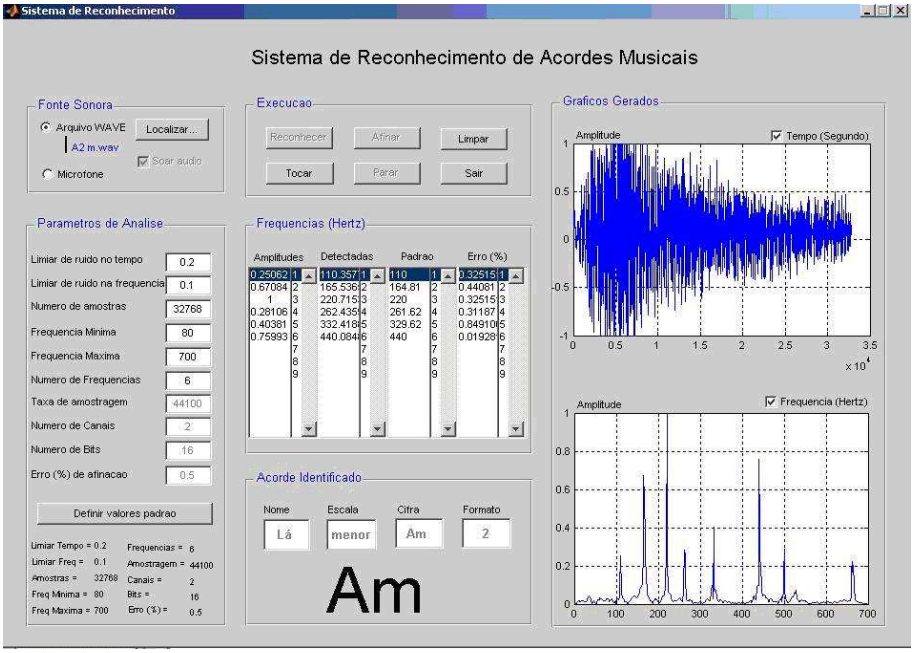


Figura 5.3: Sistema reconhecendo o acorde Lá menor em arquivo.

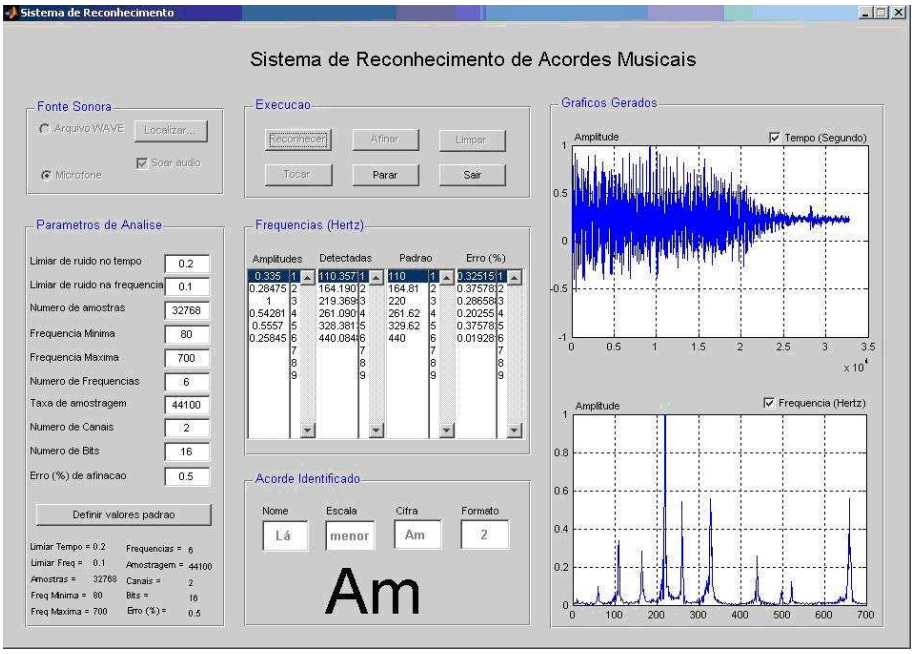


Figura 5.4: Sistema reconhecendo o acorde Lá menor.

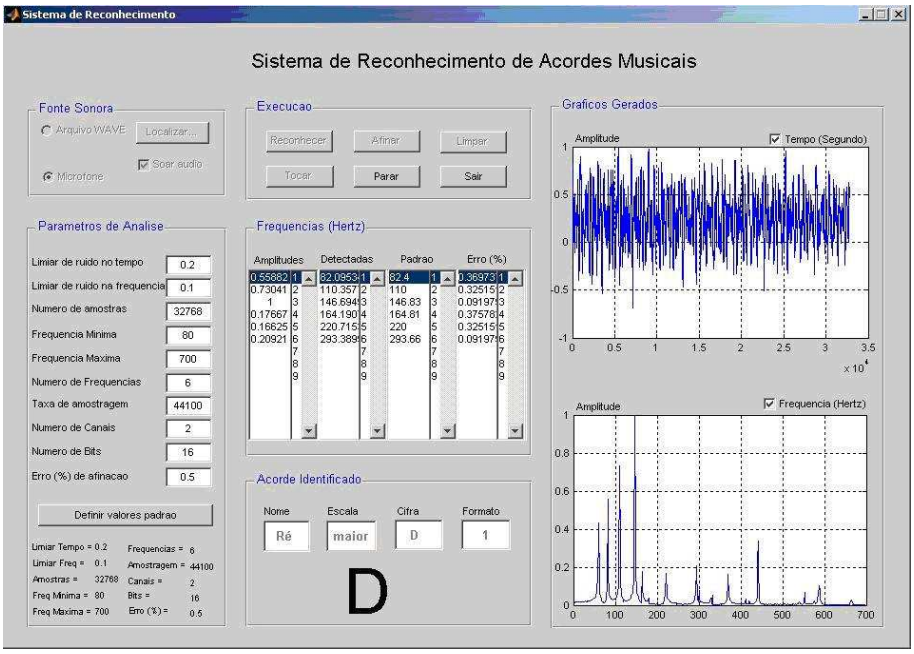


Figura 5.5: Sistema de reconhecendo o acorde Ré maior.

O artifício utilizado para reconhecer o acorde Ré maior no formato 1 pode gerar problema na detecção do acorde Ré menor também no formato 1. A diferença entre estes acordes neste formato é que a freqüência 369,99Hz do acorde maior corresponde a 349,22Hz no acorde menor. Como a fase de classificação do acorde utiliza as seis primeiras freqüências encontradas no espectro de freqüência para reconhecer o acorde, quando a intensidade da nota Mi na execução do acorde Ré menor tem valor suficiente para gerar um ou mais harmônicos, com valor de amplitude maior que o limiar do ruído na freqüência, a sexta freqüência do acorde original não será detectada, já que as seis freqüências necessárias já foram obtidas. Assim, o acorde Ré menor será reconhecido como Ré maior, pois as freqüências que os diferenciam não entram no processo de classificação. A Figura 5.6 ilustra este problema quando o Sistema reconhece o acorde Ré maior sendo que foi tocado o acorde Ré menor.

Porém, quando a nota Mi é tocada com menos intensidade gerando harmônicos com valor de amplitude menor que o limiar de ruído na freqüência, esta não gera freqüências relevantes para a fase de classificação. Desta maneira, a sexta freqüência anteriormente desprezada passa a ser considerada.

A Figura 5.7 mostra o reconhecimento correto do acorde Ré menor, onde o primeiro harmônico da nota Mi (164,190Hz) não é mais detectado e a freqüência 348,568Hz passa a ser detectada.

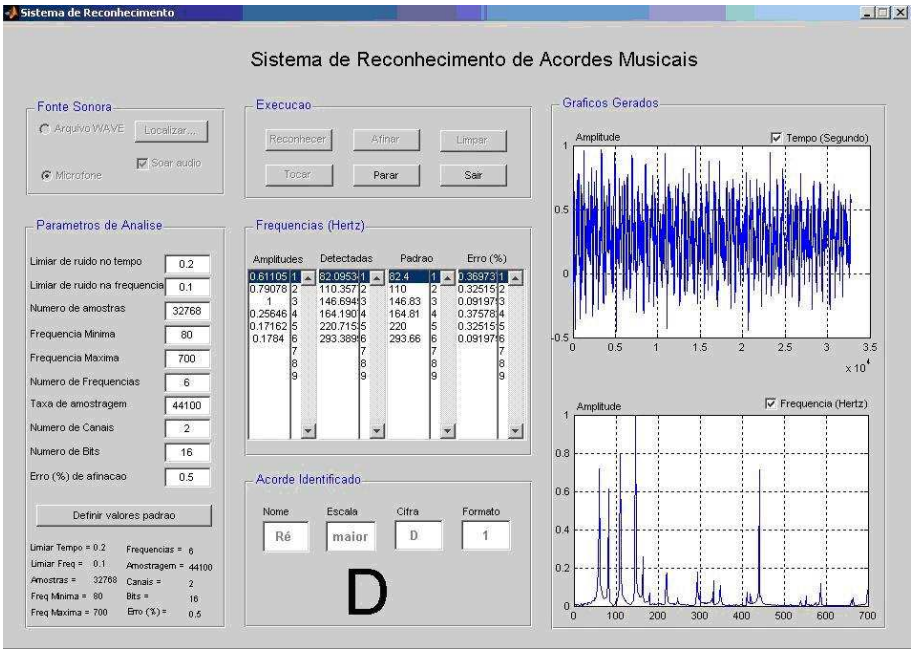


Figura 5.6: Sistema reconhecendo o acorde Ré menor como maior.

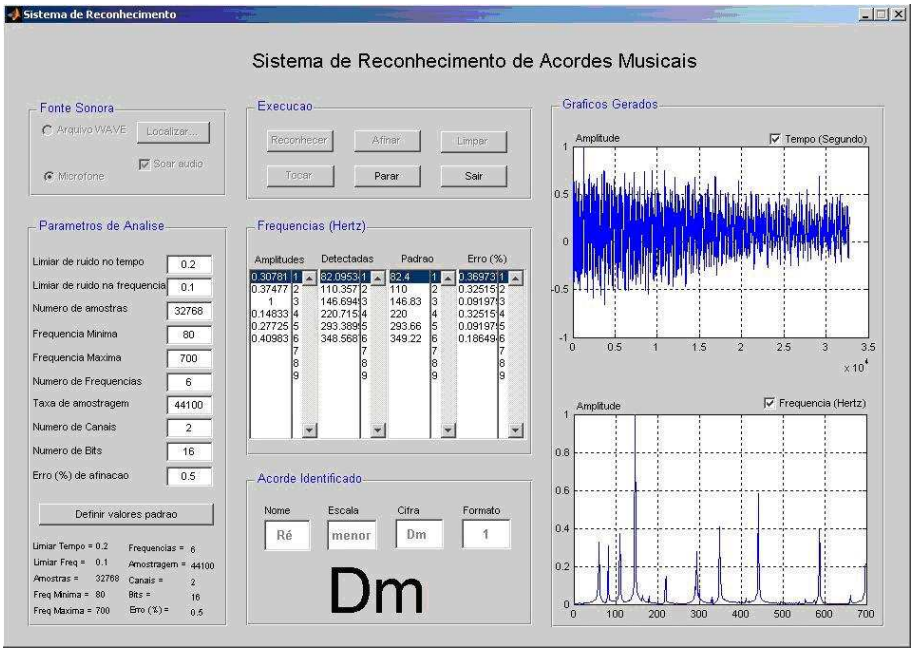


Figura 5.7: Sistema reconhecendo o acorde Ré menor corretamente.

Capítulo 6

Conclusão e Sugestões

O desafio de integrar técnicas de processamento digital de sinais às teorias de tecnologia da informação e física da música foi a motivação deste trabalho, que teve como resultado o desenvolvimento de um Sistema de Reconhecimento de acordes maiores e menores para violão elétrico em tempo real.

A primeira parte do trabalho dedicou-se a explicar de forma sucinta, mas útil, a parte da teoria musical necessária para a compreensão do funcionamento do Sistema. Adicionalmente, foram explicados os conceitos da física da música e a aplicação da transformada rápida de Fourier (FFT) ao problema proposto.

Depois do embasamento teórico estar fundamentado, foi explicado o funcionamento do Sistema e os detalhes da sua implementação. Em seguida, todos os testes realizados foram descritos e analisados. Este passo foi realizado dentro do rigor exigido pela academia, a fim de se obter resultados verdadeiros, ainda que delimitados apenas ao violão elétrico.

Os resultados acadêmicos foram muito satisfatórios, pois se observou a fidelidade do Sistema no reconhecimento em quase 100% dos casos, sendo o acorde Ré menor, no formato 1, a exceção. Esta discrepância de resultados não ocorreu devido ao processo de reconhecimento, mas sim ao formato habitual com que os músicos executam este acorde. Em qualquer outro acorde, inclusive os outros formatos (2 e 3) do Ré menor, não houve problema na identificação.

Como resultado adjacente, o Módulo de Afinação mostrou-se satisfatório em 100% dos casos. Assim, o Sistema de Reconhecimento passou a servir também como um software capaz de afinar qualquer tipo de instrumento elétrico.

A transformada rápida de Fourier, utilizada como ferramenta na transformação das amostras

de áudio do domínio do tempo para a freqüência, atendeu às expectativas, pois se mostrou bastante rápida e eficiente na execução desta tarefa. Vale ressaltar que para garantir o reconhecimento dos acordes e das notas musicais (no Módulo de Afinação) de forma satisfatória, foi necessário balancear a razão entre a taxa de amostragem e o número de amostras para atender bem o critério referente à sua resolução espectral.

Este trabalho, com seus resultados práticos, poderá servir como referência para outros que envolvam tópicos aqui relacionados, como transformada rápida de Fourier, percepção computacional da música, processamento digital de sinais, entre outros.

Como sugestão, a partir do conhecimento adquirido com a evolução desta dissertação, percebe-se ser possível ampliar o Sistema de Reconhecimento de acordes maiores e menores para violão elétrico para outros instrumentos. Outro ponto que poderá ser explorado é o desenvolvimento do Sistema em outro software que permita capturar o som em intervalos de tempo inferiores a 1 segundo. É possível também ampliá-lo para realizar o reconhecimento de outros tipos de acordes ou ainda realizar o reconhecimento conhecendo o início e fim da execução deles. Este trabalho atendeu os fins acadêmicos e a evolução deste Sistema aplicando os pontos sugeridos pode levar à sua aceitação na área comercial.

Diante dos resultados obtidos concluiu-se que é possível realizar o reconhecimento de acordes musicais maiores e menores no violão elétrico em tempo real, utilizando técnicas de DSP, através de um Sistema Especialista totalmente desenvolvido em ambiente MatLab.

A interação de técnicas de DSP, da tecnologia da informação e da física da música amplia enormemente as possibilidades de novas experiências e softwares que certamente poderão contribuir para o desenvolvimento de novos trabalhos acadêmicos.

Apêndice A

Código Fonte do Sistema de Reconhecimento de Acordes

```
%
%
%SISTEMA DE RECONHECIMENTO DE ACORDES MUSICAIS - SIRAM
%-----------------------------------------------------
%
%Utiliza a transformada rapida de Fourier, como ferramenta principal, para
%reconhecer acordes musicais.
%Parametros de entrada para a interface grafica:
%
%   LIMIAR DE RUIDO NO TEMPO -> Altura da amplitude que remove o ruido da
%                               amostra no tempo.
%   LIMIAR DE RUIDO NA FREQU -> Altura da amplitude que remove o ruido da
%                               amostra na frequencia.
%   NUMERO DE AMOSTRAS -> Quantidade de amostras de audio que sera analisada.
%   FREQUENCIA MINIMA -> Menor frequencia que sera analisada no espectro.
%   FREQUENCIA MAXIMA -> Maior frequencia que sera analisada no espectro.
%   NUMERO DE FREQUENCIAS -> Total de frequencias que serao analisadas.
%   TAXA DE AMOSTRAGEM -> Frequencia com que a amostra eh armazenada, pode
%                         receber os valores 8000, 11025, 22050 ou 44100.
%   NUMERO DE CANAIS -> Quantos canais de audio sera analisado, 1 ou 2.
%   NUMERO DE BITS -> Quantos bits sera atribuido para a amostra, 8 ou 16.
%   ERRO DE AFINACAO -> Qual a porcentagem que sera desprezada na afinacao.
%
%Desenvolvido por Sandro Alex para o projeto de conclusao do mestrado em
```

```
%Engenharia Eletrica da Universidade Federal da Bahia sob a orientacao do
%professor Doutor Antonio Cezar de Castro Lima em 2006.

%% Construcao do ambiente grafico xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function varargout = siram(varargin) gui_Singleton = 1; gui_State =
struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @siram_OpeningFcn, ...
                   'gui_OutputFcn',  @siram_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Executa quando o ambiente grafico esta abrindo xxxxxxxxxxxxxxxxxxxxxxxx
function siram_OpeningFcn(hObject, eventdata, handles, varargin)
freq = [0 0 0 0 0 0]; cifra = '   '; nome = 'ERR'; escala = '     ';
formato = 0; freq=[freq;082.40 110.00 146.83 220.00 293.66 349.22];
cifra=[cifra;'Dm ']; nome=[nome;'Ré ']; escala=[escala;'menor'];
formato=[formato;1];
%freq=[freq;082.40 110.00 146.83 220.00 293.66 369.99]; cifra=[cifra;'D  '];
freq=[freq;082.40 110.00 146.83 164.81 220.00 293.66];
cifra=[cifra;'D  ']; nome=[nome;'Ré ']; escala=[escala;'maior'];
formato=[formato;1]; freq=[freq;082.40 110.00 164.81 220.00 261.62
329.62]; cifra=[cifra;'Am ']; nome=[nome;'Lá '];
escala=[escala;'menor']; formato=[formato;1]; freq=[freq;082.40
110.00 164.81 220.00 277.18 329.62]; cifra=[cifra;'A  '];
nome=[nome;'Lá ']; escala=[escala;'maior']; formato=[formato;1];
freq=[freq;082.40 123.47 164.81 195.99 246.94 329.62];
cifra=[cifra;'Em ']; nome=[nome;'Mi ']; escala=[escala;'menor'];
formato=[formato;1]; freq=[freq;082.40 123.47 164.81 207.65 246.94
```

```
329.62]; cifra=[cifra;'E  ']; nome=[nome;'Mi '];
escala=[escala;'maior']; formato=[formato;1]; freq=[freq;082.40
130.81 164.81 195.99 261.62 329.62]; cifra=[cifra;'C  ];
nome=[nome;'Dó ']; escala=[escala;'maior']; formato=[formato;1];
freq=[freq;087.30 116.54 174.61 233.08 277.18 349.22];
cifra=[cifra;'A#m']; nome=[nome;'Lá ']; escala=[escala;'menor'];
formato=[formato;1]; freq=[freq;087.30 116.54 174.61 233.08 293.66
349.22]; cifra=[cifra;'A# ']; nome=[nome;'Lá '];
escala=[escala;'maior']; formato=[formato;1]; freq=[freq;087.30
130.81 174.61 207.65 261.62 349.22]; cifra=[cifra;'Fm '];
nome=[nome;'Fá ']; escala=[escala;'menor']; formato=[formato;1];
freq=[freq;087.30 130.81 174.61 220.00 261.62 349.22];
cifra=[cifra;'F  ']; nome=[nome;'Fá ']; escala=[escala;'maior'];
formato=[formato;1]; freq=[freq;092.49 123.47 184.99 246.94 293.66
369.99]; cifra=[cifra;'Bm ']; nome=[nome;'Si '];
escala=[escala;'menor']; formato=[formato;1]; freq=[freq;092.49
123.47 184.99 246.94 311.12 369.99]; cifra=[cifra;'B  ];
nome=[nome;'Si ']; escala=[escala;'maior']; formato=[formato;1];
freq=[freq;092.49 138.59 184.99 220.00 277.18 369.99];
cifra=[cifra;'F#m']; nome=[nome;'Fá ']; escala=[escala;'menor'];
formato=[formato;1]; freq=[freq;092.49 138.59 184.99 233.08 277.18
369.99]; cifra=[cifra;'F# ']; nome=[nome;'Fá '];
escala=[escala;'maior']; formato=[formato;1];
%freq=[freq;097.98 123.47 146.83 195.99 246.94 391.99]; cifra=[cifra;'G  '];
freq=[freq;097.98 123.47 146.83 195.99 246.94 293.66];
cifra=[cifra;'G  ']; nome=[nome;'Sol']; escala=[escala;'maior'];
formato=[formato;1]; freq=[freq;097.98 123.47 146.83 195.99 293.66
391.99]; cifra=[cifra;'G  ']; nome=[nome;'Sol'];
escala=[escala;'maior']; formato=[formato;2]; freq=[freq;097.98
130.81 195.99 261.62 311.12 391.99]; cifra=[cifra;'Cm '];
nome=[nome;'Dó ']; escala=[escala;'menor']; formato=[formato;2];
freq=[freq;097.98 130.81 195.99 261.62 329.62 391.99];
cifra=[cifra;'C  ']; nome=[nome;'Dó ']; escala=[escala;'maior'];
formato=[formato;2]; freq=[freq;097.98 146.83 195.99 233.08 293.66
391.99]; cifra=[cifra;'Gm ']; nome=[nome;'Sol'];
escala=[escala;'menor']; formato=[formato;3]; freq=[freq;097.98
146.83 195.99 246.94 293.66 391.99]; cifra=[cifra;'G  ];
```

```
nome=[nome;'Sol']; escala=[escala;'maior']; formato=[formato;3];
freq=[freq;103.82 138.59 207.65 277.18 329.62 415.30];
cifra=[cifra;'C#m']; nome=[nome;'Dó ']; escala=[escala;'menor'];
formato=[formato;2]; freq=[freq;103.82 138.59 207.65 277.18 349.22
415.30]; cifra=[cifra;'C# ']; nome=[nome;'Dó '];
escala=[escala;'maior']; formato=[formato;2]; freq=[freq;103.82
155.56 207.65 246.94 311.12 415.30]; cifra=[cifra;'G#m'];
nome=[nome;'Sol']; escala=[escala;'menor']; formato=[formato;3];
freq=[freq;103.82 155.56 207.65 261.62 311.12 415.30];
cifra=[cifra;'G# ']; nome=[nome;'Sol']; escala=[escala;'maior'];
formato=[formato;3]; freq=[freq;110.00 146.83 220.00 293.66 349.29
440.00]; cifra=[cifra;'Dm ']; nome=[nome;'Ré '];
escala=[escala;'menor']; formato=[formato;2]; freq=[freq;110.00
146.83 220.00 293.66 369.99 440.00]; cifra=[cifra;'D  '];
nome=[nome;'Ré ']; escala=[escala;'maior']; formato=[formato;2];
freq=[freq;110.00 164.81 220.00 261.62 329.62 440.00];
cifra=[cifra;'Am ']; nome=[nome;'Lá ']; escala=[escala;'menor'];
formato=[formato;2]; freq=[freq;110.00 164.81 220.00 277.18 329.62
440.00]; cifra=[cifra;'A  ']; nome=[nome;'Lá '];
escala=[escala;'maior']; formato=[formato;2]; freq=[freq;116.54
155.56 233.08 311.12 369.99 466.16]; cifra=[cifra;'D#m'];
nome=[nome;'Ré ']; escala=[escala;'menor']; formato=[formato;2];
freq=[freq;116.54 155.56 233.08 311.12 391.99 466.16];
cifra=[cifra;'D# ']; nome=[nome;'Ré ']; escala=[escala;'maior'];
formato=[formato;2]; freq=[freq;116.54 174.61 233.08 277.18 349.22
466.16]; cifra=[cifra;'A#m']; nome=[nome;'Lá '];
escala=[escala;'menor']; formato=[formato;2]; freq=[freq;116.54
174.61 233.08 293.66 349.22 466.16]; cifra=[cifra;'A# '];
nome=[nome;'Lá ']; escala=[escala;'maior']; formato=[formato;2];
freq=[freq;123.47 164.81 246.94 329.62 391.99 493.88];
cifra=[cifra;'Em ']; nome=[nome;'Mi ']; escala=[escala;'menor'];
formato=[formato;2]; freq=[freq;123.47 164.81 246.94 329.62 415.30
493.88]; cifra=[cifra;'E  ']; nome=[nome;'Mi '];
escala=[escala;'maior']; formato=[formato;2]; freq=[freq;123.47
184.99 246.94 293.66 369.99 493.88]; cifra=[cifra;'Bm '];
nome=[nome;'Si ']; escala=[escala;'menor']; formato=[formato;2];
freq=[freq;123.47 184.99 246.94 311.12 369.99 493.88];
```

```
cifra=[cifra;'B  ']; nome=[nome;'Si ']; escala=[escala;'maior'];
formato=[formato;2]; freq=[freq;130.81 174.61 261.62 349.22 415.30
523.25]; cifra=[cifra;'Fm ']; nome=[nome;'Fá '];
escala=[escala;'menor']; formato=[formato;2]; freq=[freq;130.81
174.61 261.62 349.22 440.00 523.25]; cifra=[cifra;'F  '];
nome=[nome;'Fá ']; escala=[escala;'maior']; formato=[formato;2];
freq=[freq;130.81 195.99 261.62 311.12 391.99 523.25];
cifra=[cifra;'Cm ']; nome=[nome;'Dó ']; escala=[escala;'menor'];
formato=[formato;3]; freq=[freq;130.81 195.99 261.62 329.62 391.99
523.25]; cifra=[cifra;'C  ']; nome=[nome;'Dó '];
escala=[escala;'maior']; formato=[formato;3]; freq=[freq;138.59
184.99 277.18 369.99 440.00 554.36]; cifra=[cifra;'F#m'];
nome=[nome;'Fá ']; escala=[escala;'menor']; formato=[formato;2];
freq=[freq;138.59 184.99 277.18 369.99 466.16 554.36];
cifra=[cifra;'F# ']; nome=[nome;'Fá ']; escala=[escala;'maior'];
formato=[formato;2]; freq=[freq;138.59 207.65 277.18 329.62 415.30
554.36]; cifra=[cifra;'C#m']; nome=[nome;'Dó '];
escala=[escala;'menor']; formato=[formato;3]; freq=[freq;138.59
207.65 277.18 349.22 415.30 554.36]; cifra=[cifra;'C# '];
nome=[nome;'Dó ']; escala=[escala;'maior']; formato=[formato;3];
freq=[freq;146.83 195.99 293.66 391.99 466.16 587.32];
cifra=[cifra;'Gm ']; nome=[nome;'Sol']; escala=[escala;'menor'];
formato=[formato;1]; freq=[freq;146.83 195.99 293.66 391.99 493.88
587.32]; cifra=[cifra;'G  ']; nome=[nome;'Sol'];
escala=[escala;'maior']; formato=[formato;1]; freq=[freq;146.83
220.00 293.66 349.22 440.00 587.32]; cifra=[cifra;'Dm '];
nome=[nome;'Ré ']; escala=[escala;'menor']; formato=[formato;3];
freq=[freq;146.83 220.00 293.66 369.99 440.00 587.32];
cifra=[cifra;'D  ']; nome=[nome;'Ré ']; escala=[escala;'maior'];
formato=[formato;3]; handles.banco.frequencias = freq;
handles.banco.cifras = cifra; handles.banco.nomes = nome;
handles.banco.escalas = escala; handles.banco.formatos = formato;
handles.samples = '512'; handles.limiar = '1'; handles.fMenor =
'80'; handles.fMaior = '4000'; handles.nFreq = '6';
handles.nomeArquivo = []; handles.caminhoArquivo = [];
handles.output = hObject; guidata(hObject, handles);
bLimpar_Callback(hObject, eventdata, handles);
```

```
set(handles.cbTempo,'Value',1); set(handles.cbFrequencia,'Value',1);
set(handles.tLimiarTempoR,'string','0.2');
set(handles.tLimiarFreqR,'string','0.1');
set(handles.tNumAmostrasR,'string','8000');
set(handles.tFreqMinR,'string','80');
set(handles.tFreqMaxR,'string','700');
set(handles.tNumFreqR,'string','6');
set(handles.tTaxaAmostragemR,'string','8000');
set(handles.tNumCanaisR,'string','1');
set(handles.tNumBitsR,'string','8');
set(handles.tErroAfinacao,'string','0.5');
bValores_Callback(hObject, eventdata, handles);
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Executa quando o ambiente grafico esta fechando xxxxxxxxxxxxxxxxxxxxxxxx
function varargout = siram_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Executa quando o ComboBox do grafico em funcao do tempo e' clicado xxxxx
function cbTempo_Callback(hObject, eventdata, handles)
axes(handles.aTempo); if get(handles.cbTempo,'Value') &
~isnan(handles.amplitudesTempo)
     plot(handles.amplitudesTempo); grid on;
else
    x = [0 0.01 7]; y = [1 0 0]; y = y / max(abs(y)); plot(x,y); grid off;
end set(handles.aTempo,'Default');
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Executa quando o ComboBox do grafico em funcao da frequencia e' clicado
function cbFrequencia_Callback(hObject, eventdata, handles)
axes(handles.aFrequencia); if get(handles.cbFrequencia,'Value') &
~isnan(handles.frequencias)
    plot(handles.frequencias,handles.amplitudesFrequencia); grid on;
else
    x = [0 0.01 7]; y = [1 0 0]; y = y / max(abs(y)); plot(x,y); grid off;
end set(handles.aFrequencia,'Default');
```

```
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Executa quando o botao procurar arquivo e’ clicado xxxxxxxxxxxxxxxxxxxxx
function bLocalizar_Callback(hObject, eventdata, handles) [arquivo,
caminho] = uigetfile(’*.wav’,’Escolha a amostra de audio’); if
˜isnumeric(arquivo)
    bLimpar_Callback(hObject, eventdata, handles);
    handles.nomeArquivo = arquivo;
    handles.caminhoArquivo = caminho;
    guidata(hObject,handles);
    set(handles.tArquivo,’string’,arquivo);
    set(handles.tTraco,’visible’,’on’);
    set(handles.cbSoar,’enable’,’on’);
    set(handles.bReconhecer,’enable’,’on’);
end
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Obtem a amostra de audio xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function [amostras fs amplitudes] = obterAmostra() handles =
guidata(gcbo); limiarTempo =
str2double(get(handles.eLimiarTempo,’string’)); audio =
strcat(handles.caminhoArquivo,handles.nomeArquivo); amostras =
round(str2double(get(handles.eAmostras,’string’))); [amplitudes fs
nBits opts] = wavread(audio);
set(handles.eTaxaAmostragem,’string’,num2str(fs));
set(handles.eNumBits,’string’,num2str(nBits));
set(handles.eNumCanais,’string’,num2str(opts.fmt.nChannels)); i = 1;
while abs(amplitudes(i)) < limiarTempo
    i = i + 1;
end amplitudes = amplitudes(i:length(amplitudes)); amostrasReal =
length(amplitudes); if amostras > amostrasReal, amostras =
amostrasReal;
    set(handles.eAmostras,’string’,num2str(amostrasReal));
elseif amostras < amostrasReal
    amplitudes = amplitudes(1:amostras);
end amplitudes = amplitudes/max(abs(amplitudes));
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
%% Executa quando o botao afinar e' clicado xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function bAfinar_Callback(hObject, eventdata, handles) if
~verificarValores(hObject,eventdata,handles), return; end
set(handles.bParar,'enable','on');
set(handles.bAfinar,'enable','off');
set(handles.bReconhecer,'enable','off');
set(handles.rbArquivo,'enable','off');
set(handles.rbMicrofone,'enable','off');
set(handles.eAcorde,'visible','off');
set(handles.eEscala,'visible','off');
set(handles.eCifra,'visible','off');
set(handles.eFormato,'visible','off');
set(handles.tNome,'visible','off');
set(handles.tEscala,'visible','off');
set(handles.tCifra,'visible','off');
set(handles.tFormato,'visible','off');
set(handles.pAcorde,'title','Nota Identificada');
setappdata(gcbf,'loopAfinar',true);
setappdata(gcbf,'botaoSair',false); handles = guidata(gcbo); while
getappdata(gcbf,'loopAfinar')
    % Obtem a amostra de audio
    handles.amostras = str2double(get(handles.eAmostras,'string'));
    handles.fs = str2double(get(handles.eTaxaAmostragem,'string'));
    handles.canais = str2double(get(handles.eNumCanais,'string'));
    handles.nbits = str2double(get(handles.eNumBits,'string'));
    recorder = audiorecorder(handles.fs,handles.nbits,handles.canais);
    recordblocking(recorder,1); % O matlab soh grava de 1 em 1 segundo
    y = getaudiodata(recorder);
    if (max(abs(y)) >= str2double(get(handles.eLimiarTempo,'string')))
        if (str2double(get(handles.eAmostras,'string')) > length(y))
            set(handles.eAmostras,'string',num2str(length(y)));
        end
        y = y(1:str2double(get(handles.eAmostras,'string')));
        y = y/max(abs(y)); % Normaliza
        handles.amplitudesTempo = y;
        guidata(hObject,handles);
```

```
        cbTempo_Callback(hObject, eventdata, handles);
        % Obtem o espectro de frequencias
        [handles.amplitudesFrequencia handles.frequencias] = obterFFT(hObject, eventdat
        guidata(hObject,handles);
        % Obtem os picos de frequencias
        handles.picosFrequencia = obterPicosFrequencia();
        guidata(hObject,handles);
        classificarNota();
    end
    drawnow;
end rmappdata(gcbf,'loopAfinar'); if getappdata(gcbf,'botaoSair')
close all; end
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Executa quando o botao Reconhecer e' clicado xxxxxxxxxxxxxxxxxxxxxxxxxxx
function bReconhecer_Callback(hObject, eventdata, handles) if
~verificarValores(hObject,eventdata,handles), return; end handles =
guidata(gcbo); set(handles.cbSoar,'enable','off');
set(handles.pAcorde,'title','Acorde Identificado'); if
get(handles.rbMicrofone,'value')
    set(handles.bReconhecer,'enable','off');
    set(handles.bAfinar,'enable','off');
    set(handles.bParar,'enable','on');
    set(handles.rbArquivo,'enable','off');
    set(handles.rbMicrofone,'enable','off');
end setappdata(gcbf,'loopAfinar',true);
setappdata(gcbf,'botaoSair',false); while
getappdata(gcbf,'loopAfinar')
    % Obtem a amostra de audio
    handles.amostras = str2double(get(handles.eAmostras,'string'));
    handles.fs = str2double(get(handles.eTaxaAmostragem,'string'));
    handles.canais = str2double(get(handles.eNumCanais,'string'));
    handles.nbits = str2double(get(handles.eNumBits,'string'));
    if get(handles.rbArquivo,'value')
        setappdata(gcbf,'loopAfinar',false);
        [handles.amostras handles.fs y] = obterAmostra();
    else
```

```
        recorder = audiorecorder(handles.fs,handles.nbits,handles.canais);
        recordblocking(recorder,1); % O matlab soh grava de 1 em 1 segundo
        y = getaudiodata(recorder);
    end
    if (max(abs(y)) >= str2double(get(handles.eLimiarTempo,'string')))
        if (str2double(get(handles.eAmostras,'string')) > length(y))
            set(handles.eAmostras,'string',num2str(length(y)));
        end
        y = y(1:str2double(get(handles.eAmostras,'string')));
        y = y/max(abs(y)); % Normaliza
        handles.amplitudesTempo = y;
        guidata(hObject,handles);
        cbTempo_Callback(hObject, eventdata, handles);
        % Obtem o espectro de frequencias
        [handles.amplitudesFrequencia handles.frequencias] = obterFFT(hObject, eventdat
        guidata(hObject,handles);
        % Obtem os picos de frequencias
        handles.picosFrequencia = obterPicosFrequencia();
        guidata(hObject,handles);
        % Classifica o acorde
        [handles.acordeNome handles.acordeEscala handles.acordeCifra...
            handles.acordeFormato handles.erroPorcento handles.indiceAcordes]...
            = classificarAcorde();
        guidata(hObject,handles);
        % Atualiza Tela
        set(handles.lbPicosPadrao,'string','');
        if handles.indiceAcordes
            set(handles.lbPicosPadrao,'string',...
                handles.banco.frequencias(handles.indiceAcordes,:));
        end
        set(handles.eAcorde,'string',handles.acordeNome);
        set(handles.eEscala,'string',handles.acordeEscala);
        set(handles.eCifra,'string',handles.acordeCifra);
        set(handles.eFormato,'string',num2str(handles.acordeFormato));
        set(handles.lbErro,'string',handles.erroPorcento);
        set(handles.tResultado,'string',handles.acordeCifra);
        drawnow;
```

```
    end
end if get(handles.rbArquivo,'Value')
    if get(handles.cbSoar,'Value')
        sound(handles.amplitudesTempo,handles.fs);
    end
    set(handles.bTocar,'enable','on');
    set(handles.bLimpar,'enable','on');
    set(handles.bReconhecer,'enable','off');
end rmappdata(gcbf,'loopAfinar'); if getappdata(gcbf,'botaoSair')
close all; end
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Obtem a FFT da amostra de audio solicitada xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function [amplitudesFrequencia frequencias] = obterFFT(hObject,
eventdata, handles) handles = guidata(gcbo); limiar =
str2double(get(handles.eLimiar,'string')); fmax =
str2double(get(handles.eFreqMax,'string')); fmin =
str2double(get(handles.eFreqMin,'string')); fs =
str2double(get(handles.eTaxaAmostragem,'string')); amostras =
str2double(get(handles.eAmostras,'string'));
delta = fs/amostras; % Resolução de frequencias
inicio = round(fmin*amostras/fs); indFinal =
round(fmax*amostras/fs);
fy=(0:(length(handles.amplitudesTempo-1)))*(delta); Y =
abs(fft(handles.amplitudesTempo)); freq = fy(1:round(indFinal));
ampl = Y(1:round(indFinal));
ampl = ampl/max(ampl);           % Normaliza
handles.amplitudesFrequencia = ampl; handles.frequencias = freq;
guidata(hObject,handles); cbFrequencia_Callback(hObject, eventdata,
handles);
amplitudesFrequencia = Y(inicio:indFinal); % Remove espelhamento e restringe as frequen
frequencias = fy(inicio:indFinal); % Remove espelhamento e restringe as frequencias
amplitudesFrequencia = amplitudesFrequencia/max(amplitudesFrequencia); % Normaliza
if limiar > 0 amplitudesFrequencia = amplitudesFrequencia.*(amplitudesFrequencia./limia
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Obtem os picos de frequencia xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
function picosFrequencia = obterPicosFrequencia() handles =
guidata(gcbo); numFreq =
round(str2double(get(handles.eNumFreq,'string'))); picoTemp = 0;
desce = 1; picosFrequencia = []; amplitudes = []; indfreq = 0; Y =
handles.amplitudesFrequencia; fy = handles.frequencias; for i =
1:length(fy)
    if (Y(i) >= picoTemp)     % O valor da frequencia esta crescendo
        desce = 0;
    else
        if (desce ~= 1)       % Começou a descer, provável pico
            if (indfreq > 0)  % Se ja existe alguma frequencia alocada
                if ((fy(i-1) - picosFrequencia(indfreq) > ...
                        picosFrequencia(indfreq) * 1.0594631...
                        - picosFrequencia(indfreq)))
                    picosFrequencia = [picosFrequencia fy(i-1)];
                    amplitudes = [amplitudes Y(i-1)];
                    indfreq = indfreq + 1;
                else
                   if (Y(i-1) > amplitudes(indfreq))
                       picosFrequencia(indfreq) = fy(i-1);
                       amplitudes(indfreq) = Y(i-1);
                   end
                end
            else
                picosFrequencia = [picosFrequencia fy(i-1)];
                amplitudes = [amplitudes Y(i-1)];
                indfreq = indfreq + 1;
            end
            desce = 1;
        end
    end
    picoTemp = Y(i);
end if (indfreq > numFreq)
    picosFrequencia = picosFrequencia(1:numFreq);
    amplitudes = amplitudes(1:numFreq);
end
set(handles.lbPicosDetectados,'string',num2str(picosFrequencia'));
```

```
set(handles.lbAmpDetectados,'string',num2str(amplitudes'));
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Classifica nota nos picos de frequencia xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function classificarNota() handles = guidata(gcbo); frequencia =
handles.picosFrequencia(1); erro =
str2double(get(handles.eErro,'string')); notaFre=[32.7 34.6 36.7
38.8 41.2 43.6 46.2 48.9 51.9 55 58.2 61.7 65.4]; notaCif=['C
';'C#';'D ';'D#';'E ';'F ';'F#';'G ';'G#';'A ';'A#';'B ';'C '];
oitava = 1; f = 1; while(frequencia > notaFre(13))
    frequencia = frequencia / 2;
    oitava = oitava + 1;
end if (frequencia < notaFre(1)) return; end while (abs(frequencia -
notaFre(f)) > abs(frequencia - notaFre(f+1)))
    f = f + 1;
    if  (f == 13) break; end
end dif = frequencia - notaFre(f); if (dif > 0) &
((notaFre(f)+(notaFre(f)*erro/100)) < frequencia)
    set(handles.tAjuste,'string','FOLGAR');
elseif (dif < 0) & ((notaFre(f)-(notaFre(f)*erro/100)) > frequencia)
    set(handles.tAjuste,'string','APERTAR');
else
    set(handles.tAjuste,'string','');
end set(handles.tResultado,'string',notaCif(f,:));
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Classifica o acorde xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function [an ae ac af erro lin] = classificarAcorde() handles =
guidata(gcbo); an = ''; ae = ''; ac = ''; af = ''; lin = 0; erro =
[]; if length(handles.picosFrequencia) < str2num(handles.nFreq)
    if (get(handles.rbArquivo,'value'))
        errordlg(['A amostra possui menos de ' num2str(handles.numFreq) ' picos de fre
    end
    return;
end acordes = handles.banco.frequencias; picosFrequencia =
handles.picosFrequencia; [linhas colunas] = size(acordes); inicio =
1; lin = inicio; fim = linhas; achou = 1; for col = 1:colunas
```

```
    % Enquanto estiver na faixa de frequencias provaveis
    while (lin < fim)
        % Se a diferenca da proxima frequencia for menor que a atual
        if (abs((acordes(lin+1,col)-picosFrequencia(col)))) <= ...
                (abs((acordes(lin,col)-picosFrequencia(col))))
            if acordes(inicio,col)~=acordes(lin+1,col)
                inicio = lin + 1;
            end
            lin = lin + 1;
        else
            break;
        end
    end
    % Verifica se a provavel frequencia identificada eh a correta
    if (abs(acordes(lin,col)*1.0594631-picosFrequencia(col)) < ...
            abs(acordes(lin,col)-picosFrequencia(col))) | ...
            (abs(acordes(lin,col)/1.0594631-picosFrequencia(col)) < ...
            abs(acordes(lin,col)-picosFrequencia(col)))
        achou = 0;
        lin = 0;
        break;
    end
    fim = lin;
    lin = inicio;
end if (achou)
    for i = 1:length(handles.banco.frequencias(lin,:))
        freqDetec = handles.picosFrequencia(i);
        freqPadrao = handles.banco.frequencias(lin,i);
        erro = [erro abs(((freqDetec-freqPadrao)/freqPadrao)*100)];
    end
    an = handles.banco.nomes(lin,:);
    ae = handles.banco.escalas(lin,:);
    ac = handles.banco.cifras(lin,:);
    af = handles.banco.formatos(lin);
end
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
%% Verifica os valores no ambiente grafico xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function result = verificarValores(hObject, eventdata, handles)
handles = guidata(gcbo); limiarF =
str2double(get(handles.eLimiar,'string')); limiarT =
str2double(get(handles.eLimiarTempo,'string')); amostras =
round(str2double(get(handles.eAmostras,'string'))); fmin =
str2double(get(handles.eFreqMin,'string')); fmax =
str2double(get(handles.eFreqMax,'string')); numFreq =
str2double(get(handles.eNumFreq,'string')); nbits =
str2double(get(handles.eNumBits,'string')); canais =
str2double(get(handles.eNumCanais,'string')); erroAfinacao =
str2double(get(handles.eErro,'string')); fs =
str2double(get(handles.eTaxaAmostragem,'string')); result = 0;
samples = handles.samples; limiar = handles.limiar; fMenor =
handles.fMenor; fMaior = handles.fMaior; nFreq =
str2num(handles.nFreq); if isnan(limiarF)| isnan(limiarT)|
isnan(amostras)| isnan(fmin)|...
        isnan(fmax)| isnan(numFreq)| isnan(nbits)| isnan(canais)|...
        isnan(fs)| isnan(erroAfinacao)
    errordlg('Todos os campos devem ter valores numericos','Erro');
    return;
end if limiarF < 0 | limiarF > str2num(limiar) | limiarT < 0 |
limiarT > str2num(limiar)
    errordlg(['O limiar de ruido deve estar entre 0 e ' limiar],'Erro');
    return;
end if amostras < str2num(samples)
    errordlg(['O numero de amostras tem de ser maior que ' samples],'Erro');
    return;
end if fmin < 1 | fmin > str2num(fMenor)
    errordlg(['A frequencia minima deve estar entre 1 e ' fMenor ' Hz'],'Erro');
    return;
end if fmax < 700 | fmax > str2num(fMaior)
    errordlg(['A frequencia maxima deve estar entre 700 e ' fMaior ' Hz'],'Erro');
    return;
end if numFreq < nFreq
    errordlg('O numero de frequencias tem de ser maior que 5','Erro');
    return;
```

```
end if rem(amostras,fix(amostras)) ~= 0 | rem(fmin,fix(fmin)) ~= 0 |
...
        rem(fmax,fix(fmax)) ~= 0 | rem(numFreq,fix(numFreq)) ~= 0
    errordlg('Os campos devem ter valores inteiros, exceto o limiar','Erro');
    return
end if get(handles.rbArquivo,'value')
    if length(handles.nomeArquivo) == 0
        errordlg('Eh necessario selecionar um arquivo','Erro');
        return;
    end
    try
        [amplitudes fs nBits] = wavread(strcat(handles.caminhoArquivo,handles.nomeArqui
    catch
        errordlg('O arquivo nao eh do tipo WAVE ou qtd amostras muito grande','Erro');
        return
    end
end if erroAfinacao < 0 | erroAfinacao > 100
    errordlg('O erro de afinacao deve estar entre 0 e 100','Erro');
    return;
end if nbits ~= 8 & nbits ~= 16 & nbits ~= 24
    errordlg('O numero de bits tem de ser 8, 16 ou 24','Erro');
    return
end if canais ~= 1 & canais ~= 2
    errordlg('O numero de canais tem de ser 1 ou 2','Erro');
    return
end if fs ~= 8000 & fs ~= 11025 & fs ~= 22050 & fs ~= 44100
    errordlg('A taxa de amostragem tem de ser 8000, 11025, 22050 ou 44100 Hz','Erro');
    return
end handles.limiarTempo =  limiarT; handles.limiarFreq =  limiarF;
handles.amostras = amostras; handles.fmin = fmin; handles.fmax =
fmax; handles.numFreq = numFreq; handles.fs = fs; handles.canais =
canais; handles.nbits = nbits; guidata(hObject,handles); result = 1;
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Executa quando o botao tocar e' clicado xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function bTocar_Callback(hObject, eventdata, handles) handles =
guidata(gcbo); if length(handles.amplitudesTempo)
```

```
    sound(handles.amplitudesTempo,handles.fs);
else
    errordlg('Nao existe uma amostra de audio para ser tocada!');
end
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Funcoes de controle da tela xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
function TelaPrincipal_CloseRequestFcn(hObject, eventdata, handles)
delete(hObject);

function bSair_Callback(hObject, eventdata, handles) if
~isappdata(gcbf,'loopAfinar')
    delete(gcbf)
else
    setappdata(gcbf,'loopAfinar',false);
    setappdata(gcbf,'botaoSair',true);
end

function bParar_Callback(hObject, eventdata, handles)
set(handles.bReconhecer,'enable','on');
set(handles.bAfinar,'enable','on');
set(handles.bParar,'enable','off');
%set(handles.bSair,'enable','on');
set(handles.rbArquivo,'enable','on');
set(handles.rbMicrofone,'enable','on'); if
get(handles.rbMicrofone,'value')
    set(handles.tTraco,'visible','on');
end if strcmp(get(handles.tNome,'visible'),'off')
    set(handles.eAcorde,'visible','on');
    set(handles.eEscala,'visible','on');
    set(handles.eCifra,'visible','on');
    set(handles.eFormato,'visible','on');
    set(handles.tNome,'visible','on');
    set(handles.tEscala,'visible','on');
    set(handles.tCifra,'visible','on');
    set(handles.tFormato,'visible','on');
    set(handles.pAcorde,'title','Nota/Acorde Identificado');
```

```
end bLimpar_Callback(hObject, eventdata, handles);
setappdata(gcbf,'loopAfinar',false);

function pFonte_SelectionChangeFcn(hObject, eventdata, handles)
bLimpar_Callback(hObject, eventdata, handles); switch
get(hObject,'Tag')
    case 'rbArquivo'
        set(handles.bLocalizar,'enable','on');
        set(handles.bReconhecer,'enable','off');
        set(handles.bAfinar,'enable','off');
        set(handles.eTaxaAmostragem,'enable','off');
        set(handles.eNumCanais,'enable','off');
        set(handles.eNumBits,'enable','off');
        set(handles.eErro,'enable','off');
    case 'rbMicrofone'
        set(handles.bLocalizar,'enable','off');
        set(handles.bReconhecer,'enable','on');
        set(handles.bAfinar,'enable','on');
        set(handles.eTaxaAmostragem,'enable','on');
        set(handles.eNumCanais,'enable','on');
        set(handles.eNumBits,'enable','on');
        set(handles.eErro,'enable','on');
        set(handles.bTocar,'enable','off');
        set(handles.cbSoar,'value',1);
        set(handles.cbSoar,'enable','off');
end

function bValores_Callback(hObject, eventdata, handles)
set(handles.eLimiarTempo,'string',get(handles.tLimiarTempoR,'string'));
set(handles.eLimiar,'string',get(handles.tLimiarFreqR,'string'));
set(handles.eAmostras,'string',get(handles.tNumAmostrasR,'string'));
set(handles.eFreqMin,'string',get(handles.tFreqMinR,'string'));
set(handles.eFreqMax,'string',get(handles.tFreqMaxR,'string'));
set(handles.eNumFreq,'string',get(handles.tNumFreqR,'string'));
set(handles.eTaxaAmostragem,'string',get(handles.tTaxaAmostragemR,'string'));
set(handles.eNumCanais,'string',get(handles.tNumCanaisR,'string'));
set(handles.eNumBits,'string',get(handles.tNumBitsR,'string'));
```

```
set(handles.eErro,'string',get(handles.tErroAfinacao,'string'));

function bLimpar_Callback(hObject, eventdata, handles) if
strcmp(get(handles.tTraco,'visible'),'off') & ...
        strcmp(get(handles.tResultado,'string'),''), return; end
set(handles.bLimpar,'enable','off');
set(handles.bTocar,'enable','off'); handles.limiarTempo = 0;
handles.limiarFreq = 0; handles.amostras = 0; handles.fmin = 0;
handles.fmax = 0; handles.numFreq = 0; handles.fs = 0;
handles.canais = 0; handles.nbits = 0; handles.picosFrequencia = [];
handles.frequencias = []; handles.erroPorcento = [];
handles.amplitudesFrequencia = []; handles.amplitudesTempo = [];
handles.indiceAcordes = []; handles.acordeNome = '';
handles.acordeEscala = ''; handles.acordeCifra = '';
handles.acordeFormato = ''; handles.nomeArquivo = '';
handles.caminhoArquivo = ''; guidata(hObject, handles);
axes(handles.aTempo); x = [0 0.01 7]; y = [1 0 0]; y = y /
max(abs(y)); plot(x,y); grid off; set(handles.aTempo,'Default');
axes(handles.aFrequencia); x = [0 0.01 7]; y = [1 0 0]; y = y /
max(abs(y)); plot(x,y); grid off;
set(handles.aFrequencia,'Default');
set(handles.lbPicosDetectados,'string','');
set(handles.lbPicosPadrao,'string','');
set(handles.lbAmpDetectados,'string','');
set(handles.lbErro,'string',''); set(handles.eAcorde,'string','');
set(handles.eEscala,'string',''); set(handles.eCifra,'string','');
set(handles.eFormato,'string','');
set(handles.tResultado,'string','');
set(handles.eFormato,'string',''); set(handles.tAjuste,'string','');
set(handles.tArquivo,'string','');
set(handles.tTraco,'visible','off');
%xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

%% Funcoes que devem estar instanciadas para o bom funcionamento xxxxxxxxxx
function eFormato_CreateFcn(hObject, eventdata, handles) if ispc
set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end function eCifra_CreateFcn(hObject, eventdata, handles) if ispc
set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eAcorde_CreateFcn(hObject, eventdata, handles) if ispc,
set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function lbErro_CreateFcn(hObject, eventdata, handles) if ispc,
set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function lbPicosPadrao_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function lbPicosDetectados_CreateFcn(hObject, eventdata,
handles) if ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eErro_CreateFcn(hObject, eventdata, handles) if ispc
set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eNumCanais_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eNumBits_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eTaxaAmostragem_CreateFcn(hObject, eventdata, handles)
if ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eLimiarTempo_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eFreqMax_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eNumFreq_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eLimiar_CreateFcn(hObject, eventdata, handles) if ispc,
```

```
set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eAmostras_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eFreqMin_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function eEscala_CreateFcn(hObject, eventdata, handles) if ispc,
set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function lbAmpDetectados_CreateFcn(hObject, eventdata, handles)
if ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function listbox6_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function listbox7_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function listbox8_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function listbox9_CreateFcn(hObject, eventdata, handles) if
ispc, set(hObject,'BackgroundColor','white'); else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end function lbPicosDetectados_Callback(hObject, eventdata, handles)
function lbPicosPadrao_Callback(hObject, eventdata, handles)
function lbAmpDetectados_Callback(hObject, eventdata, handles)
function lbErro_Callback(hObject, eventdata, handles) function
eLimiarTempo_Callback(hObject, eventdata, handles) function
eLimiar_Callback(hObject, eventdata, handles) function
eAmostras_Callback(hObject, eventdata, handles) function
eFreqMin_Callback(hObject, eventdata, handles) function
eFreqMax_Callback(hObject, eventdata, handles) function
eNumFreq_Callback(hObject, eventdata, handles) function
eTaxaAmostragem_Callback(hObject, eventdata, handles) function
```

```
eNumCanais_Callback(hObject, eventdata, handles) function
eNumBits_Callback(hObject, eventdata, handles) function
eErro_Callback(hObject, eventdata, handles) function
cbSoar_Callback(hObject, eventdata, handles) function
bReconhecer_KeyPressFcn(hObject, eventdata, handles) function
bReconhecer_ButtonDownFcn(hObject, eventdata, handles) function
listbox6_Callback(hObject, eventdata, handles) function
listbox7_Callback(hObject, eventdata, handles) function
listbox8_Callback(hObject, eventdata, handles) function
listbox9_Callback(hObject, eventdata, handles)
```

Referências Bibliográficas

[Alv06] José Rodríguez Alvira, *Music theory web*, http://www.teoria.com/, Agosto 2006.

[Bac97] John Backus, *The acoustical foundations of music*, 2ª ed., W.W. Norton & Company, New York, 1997.

[Bor98] Signal Processing Bores, *Training in dsp and media processing*, http://www.bores.com, 1998.

[DJ97] Charles Dodge and Thomas A. Jerse, *Computer music synthesis, composition, and performance*, 2ª ed., Schirmer Books, New York, 1997.

[Dov99] João Cândido Lima Dovicchi, *Novos coeficientes wavelets baseados em intervalos musicais para análise de timbres de instrumentos acústicos*, Ph.D. thesis, Universidade Federal de Uberlândia, Minas Gerais, 1999.

[Fre94] Ira M. Freeman, *Som e ultra-som*, 2ª ed., Distribuidora Record, Rio de Janeiro, 1994.

[HV01] Simon Haykin and Barry Van Veen, *Sinais e sistemas*, 4ª ed., Bookman, Porto Alegre, 2001.

[KD00] Stefan Kostka and Payne Dorothy, *Tonal harmony, with a introduction to twentieth-century music*, 4ª ed., McGraw-Hill, 2000.

[Kla04] Anssi Klapuri, *Signal processing methods for the automatic transcription of music*, Ph.D. thesis, Tampere University of Technology, Tampere, 2004.

[Lac96] Osvaldo Lacerda, *Compêndio de teoria elementar da música*, 9ª ed., Ricordi Brasileira S.A., 1996.

[Mac01] A. C. Machado, *Tradutor de arquivos midi para texto utilizando linguagem funcional clean*, 2001.

[Mar03] Sylvain Marchand, *An efficient pitch-tracking algorithm using a combination of fourier transforms*, The cost G-6 conference on digital audio effects (2003).

[Med97] Bohumil Med, *Teoria da música*, 4ª ed., Musimed, Brasília, DF, 1997.

[Moo90] F. Richard Moore, *Elements of computer music*, Prentice Hall, Englewood Cliffs, 1990.

[MQ05] Adriano Mitre and Marcelo Queiroz, *Um sistema automático de transcrição melódica*, Simpósio Brasileiro de Computação Musical (2005).

[Nag03] Keerthi Nagaraj, *Toward automatic transcription - pitch tracking in polyphonic*, EE381K- Multidimensional Digital Signal Processing (2003).

[Net06] Luiz Netto, *Matemática na música*, 2006.

[OJ02] Luis Ortiz and F. Javier, *Polyphonic transcription using piano modeling for spectral pattern recognition*, International Conference on Digital Audio Effects (2002).

[Ols67] Harry F. Olson, *Music, physics and engineering*, 2ª ed., Dover Publications, New York, 1967.

[Org75] International Standard Organization, *Acoustics - standard tuning frequency*, http://www.iso.org, 1975.

[OS89] Alan V. Oppenheim and Ronald W. Schafer, *Signals and systems*, Prentice-Hall, NJ, USA, 1989.

[Smi99] Steve W. Smith, *The scientist and engineer's guide to digital signal processing*, 2ª ed., California Technical Publishing, California, 1999.

[Ste96] Ken Steiglitz, *A digital signal processing primer, with applications to digital audio and computer music*, Menio Park (1996).

[Wat94] John Watkinson, *An introduction to digital audio*, Focal Press, 1994.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas