UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Monitoramento da Rugosidade no Processo de Torneamento Duro Utilizando Coeficientes Mel-Cepstrais de Sinais Acústicos e Modelos de Misturas de Gaussianas

Edielson Prevato Frigieri

Itajubá, Agosto de 2013

UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Edielson Prevato Frigieri

Monitoramento da Rugosidade no Processo de Torneamento Duro Utilizando Coeficientes Mel-Cepstrais de Sinais Acústicos e Modelos de Misturas de Gaussianas

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Produção como parte dos requisitos para a obtenção do Título de ***Mestre em Ciências em Engenharia de Produção***

Orientador: Prof. Anderson P. Paiva, Dr.
Co-orientador: Prof. João Roberto Ferreira, Dr.

Agosto de 2013

Itajubá – MG

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

EDIELSON PREVATO FRIGIERI

MONITORAMENTO DA RUGOSIDADE NO PROCESSO DE TORNEAMENTO DURO UTILIZANDO COEFICIENTES MEL-CEPSTRAIS DE SINAIS ACÚSTICOS E MODELOS DE MISTURAS DE GAUSSIANAS

Dissertação aprovada por banca examinadora em 26 de junho de 2013, conferindo ao autor o título de ***Mestre em Ciências em Engenharia de Produção.***

Banca Examinadora:
Prof. Anderson P. Paiva, Dr. (Orientador)
Prof. João Roberto Ferreira, Dr. (Co-orientador)
Prof. Carlos Alberto Ynoguti, Dr.
Prof. Rafael Coradi Leme, Dr.

Itajubá 2013

DEDICATÓRIA

Aos meus pais, Edi e Suely, responsáveis pela minha formação como pessoa e profissional e a minha esposa Karina, pelo apoio total e pelo tempo que deixamos de estar juntos...

AGRADECIMENTOS

À minha esposa Karina que dedicou tempos preciosos para revisar o trabalho e dar opiniões. Que me apoiou e incentivou em todas as dificuldades encontradas.

À minha família pelo apoio e por sempre incentivarem meus estudos, em especial meu pai Edi, minha mãe Suely e minha segunda mãe Marli.

Ao Prof. Dr. Anderson Paulo de Paiva por abrir às portas e permitir que eu finalizasse um sonho, além das orientações e paciência ao longo do desenvolvimento deste trabalho.

Ao Prof. Dr. Carlos Alberto Ynoguti que nunca deixou de me apoiar e ajudar, mesmo nos momentos mais difíceis, e que sempre acreditou que terminaríamos este trabalho.

Aos meus colegas de pós-graduação, em especial Paulo Henrique Campos, Rogério Peruchi, Luis Gustavo e Tarcísio, que trabalharam muito para tornarem o trabalho possível.

A todos, muito obrigado!

“Existe apenas um bem, o saber, e apenas um mal, a ignorância.”

Sócrates.

Resumo

O objetivo deste trabalho é avaliar a emissão de sinais acústicos durante o processo de torneamento do aço ABNT 52100 endurecido e identificar os parâmetros do sinal que sejam capazes de representar a rugosidade resultante da peça. Partindo desses parâmetros, propõe-se um novo método de monitoramento do processo utilizando modelos de misturas de Gaussianas (GMM). A principal característica desta nova abordagem é a utilização do som como meio de monitoramento. Além de apresentar vantagens como a facilidade de aquisição de dados e o baixo custo dos sensores (microfones), ainda é um sinal pouco explorado em técnicas de monitoramento. Para correlacionar a rugosidade resultante do processo de torneamento com os parâmetros do som emitido, técnicas de processamento de sinais digitais foram utilizadas para a extração de parâmetros como perfil de energia e coeficientes mel-cepstrais (MFCC). Estes parâmetros foram utilizados para treinar modelos de misturas de Gaussianas para representar cada grupo de rugosidade identificado. No treinamento dos modelos GMM, foram utilizadas 4 e 8 gaussianas a fim de avaliar o impacto no desempenho do método proposto. Ao apresentar o conjunto de sinais de teste aos modelos treinados, obteve-se uma superioridade nos modelos treinados com MFCC, apresentando uma taxa de acerto média de 94,27%. Já o número de Gaussianas não proporcionou aumento significativo de desempenho. A partir destes resultados, concluiu-se que o sinal acústico (som), através de parâmetros como os coeficientes mel-cepstrais, pode ser utilizado como forma de identificar variações no comportamento do processo, principalmente a rugosidade resultante da peça, o que permite o monitoramento da qualidade do processo de forma não destrutiva.

Palavras-chave: Torneamento duro, Processamento de sinais digitais, Coeficientes Mel-Cepstrais, Emissão acústica, Som, Modelo de Mistura de Gaussianas.

Abstract

The purpose of this work is to evaluate the acoustic signal emission during the AISI 52100 steel hard turning process and identify the parameters in the signal that are able to represent the surface roughness of the workpiece resulting from the process. Based on these parameters, propose a new method for monitoring the processes based on Gaussian mixture models (GMM). The main characteristic of this new approach is the application of the sound as a means for monitoring. Besides presenting advantages as the data acquisition facility and the low cost for the sensors (microphones), is still a signal largely unexplored in monitoring techniques. For correlating the roughness resulting from the turning process with the parameters of the emitted sound, digital signal processing techniques were used for extracting parameters like energy profile and Mel-frequency cepstral coefficients (MFCC). These parameters were used for training the Gaussian mixture models to represent each roughness group that was identified. For modeling of GMM's, were used 4 and 8 Gaussians to evaluate the impact in the performance of the proposed method. Presenting the set of test signals to the trained models, obtained a superiority of the model trained with MFCC that presented a performance of 94.27%. The number of Gaussians did not present significant rise in the performance. Based on these results, it is concluded that the acoustic signal (sound), using parameters as the MFCC's, can be used as a way to identify variation in the behavior of the process, mainly the roughness of the workpiece resulting from the turning process, what permits the quality monitoring of the process in a nondestructive way.

Keywords: Hard turning, Digital signal processing, Mel-frequency cepstrals coefficients, Acoustic Emission, Sound, Gaussian mixture models.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MFCC	*Mel-Frequency Cepstral Coefficients*
GMM	*Gaussian Mixture Model*
EA	Emissão Acústica
DSP	*Digital Signal Processing*
FFT	*Fast Fourier Transform*
HMM	*Hidden Markov Models*
ANN	*Artificial Neural Networks*
EM	*Expectation Maximization*
ABNT	Associação Brasileira de Normas Técnicas
CCD	Arranjo Composto Central ou Box-Wilson *Central Composite Desig*
CNC	Comando Numérico Computadorizado
HRC	Rockwell escala C

LISTA DE SÍMBOLOS

Hz	Unidade para frequência (Hertz)
MHz	mega (10^6) Hertz
kHz	kilo (10^3) Hertz
ms	mili (10^{-3}) segundos
kW	kilo (10^3) Watt
π	Letra grega que representa o valor 3,141516
ln	Logaritmo natural ou neperiano
$\sum$	Somatório
$\prod$	Produtório
a_p	Profundidade de corte
f	Taxa de avanço
V_c	Velocidade de corte
λ	Letra grega lambda
μ	Letra grega um
σ	Letra grega sigma
R_a	Rugosidade média aritmética
R_q	Rugosidade média quadrática
R_t	Rugosidade total
R_y	Rugosidade máxima
R_z	Rugosidade média
rpm	Rotações por minuto

SUMÁRIO

1. Introdução

1.1 Considerações Iniciais

O processo de torneamento em aço duro tem sido amplamente estudado por apresentar mais benefícios que os processos tradicionais. Um dos principais benefícios é a eficiência na redução do tempo de processo em cada operação, que só pode ser alcançada com valores adequados dos parâmetros do processo e escolha adequada do material da ferramenta e sua geometria (PAIVA *et al.*, 2012).

Após determinar o ponto ótimo de operação do processo, material e geometria da ferramenta, o grande desafio é conseguir manter o equipamento operando com estes parâmetros ou mesmo identificar quando houver alterações nas características de saída, como por exemplo, a rugosidade da peça usinada. O próprio desgaste da ferramenta provoca uma alteração na rugosidade, o que o torna um parâmetro importante na identificação do momento adequando para a troca da ferramenta e manutenção do ponto ótimo (BONIFÁCIO E DINIZ, 1994).

A fim de garantir a operação no ponto ótimo, diferentes métodos de monitoramento tem sido objeto de estudo de trabalhos científicos (SICK, 2002; ZHU, WONG E HONG, 2009). Estes métodos foram classificados como diretos (ótico, radioativo, elétrico resistivo, etc.) e indiretos (emissão acústica, corrente do motor, força de corte, vibração, etc.). Os trabalhos mais recentes têm concentrado os esforços nos métodos indiretos (BHASKARAN *et al.*, 2012; LI, 2002), que utilizam meios de medição não intrusivos e que não necessitam que o processo pare para realizar medições, possibilitando ajustes em tempo real. O foco deste trabalho está nos métodos indiretos por apresentarem um real ganho de produtividade ao permitir que seja monitorada a qualidade do processo sem que haja interferência no mesmo.

Dentro dos métodos indiretos, a emissão acústica (EA) tem tido destaque por ser um sinal com faixa de frequência acima dos demais sinais de vibração da máquina e ruídos do ambiente (XI, HAN E LIU, 2010). Podem-se encontrar diversas propostas de monitoramento utilizando emissão acústica para o monitoramento de diferentes processos de usinagem como retificação, fresamento (JEMIELNIAK E ARRAZOLA, 2008), furação (ANDOH, DAVIS E ANTONIO, 2008), torneamento (SCHEFFER *et al.*, 2003), entre outros.

Baseando-se nos métodos indiretos e principalmente nos métodos que utilizam EA, uma nova abordagem será apresentada neste trabalho que é utilizar como método indireto, os sinais acústicos (sons) emitidos durante o processo de torneamento duro, que ainda são pouco explorados no monitoramento de processos. Este tipo de sinal, encontra-se na faixa de

frequência audível (20 Hz e 20.000 Hz), o que possibilita a utilização de um microfone como meio de captura, tornando o sistema mais simples e com custo menor por não necessitar de sensores especiais, como acelerômetros ou dinamômetros normalmente utilizados para sinais de EA.

Portanto, a proposta deste trabalho é capturar e analisar o som emitido durante o processo de torneamento do aço ABNT 52100 endurecido, identificar qual parâmetro melhor representa o fenômeno e correlacioná-lo com a rugosidade da peça. Serão extraídos o perfil de energia e os MFCCs (*Mel-Frequency Cepstral Coefficients*) do sinal capturado. Cada parâmetro treinará um modelo de classificação baseado em GMM (*Gaussian Mixture Model*) que, posteriormente, será testado para verificar qual o nível de correlação com o grupo de rugosidade que o treinou. A partir do parâmetro mais representativo, será proposto um método indireto de monitoramento da rugosidade, baseado no sinal acústico capturado durante o processo de usinagem.

1.2 Objetivo

O objetivo principal deste trabalho é propor um novo método de monitoramento indireto baseado no som, emitido durante o processo de torneamento duro, e utilizando uma técnica baseada em MFCC e modelos de misturas de gaussianas (GMM).

Podem também ser relacionados como objetivos deste trabalho:

- Identificar qual parâmetro extraído do sinal acústico possui maior correlação com a rugosidade resultante do processo de torneamento, comparando o MFCC e o perfil de energia;
- Validar a utilização do som emitido no processo como método indireto de monitoramento da rugosidade, e, consequentemente da qualidade do produto.

Como objetivos secundários, pode-se destacar:

- Comparar o desempenho do método proposto utilizando 4 e 8 gaussianas para o modelamento dos GMMs;
- Desenvolver um software para efetuar as etapas de extração de parâmetros e modelamento estatístico do método proposto.

1.3 Estrutura do Trabalho

A estrutura do trabalho está dividida conforme resumo a seguir.

O capitulo 2 apresentará a revisão bibliográfica referente aos temas utilizados no desenvolvimento desta pesquisa, apresentando trabalhos científicos já publicados e que sejam relacionados aos assuntos abordados. Também será apresentado neste capitulo o método de processamento de sinais digitais, que envolve a extração de parâmetros do sinal e método de modelamento baseado em GMM.

O capitulo 3 abordará o planejamento do experimento. Serão detalhados os equipamentos utilizados assim como a montagem da base de dados utilizada na avaliação do método proposto.

O capitulo 4 apresentará os resultados dos experimentos realizados, correlacionando os parâmetros do som capturado com a rugosidade resultante.

O capitulo 5 apresentará as conclusões e análises do trabalho, baseado nos resultados obtidos no capítulo anterior. Sugestões para trabalhos futuros serão mencionados neste capítulo.

2. Revisão de Literatura

2.1 Métodos de Monitoramento de Processos de Fabricação

Devido ao aumento na competitividade, as empresas precisam cada vez mais se preocuparem com a rápida mudança das tecnologias e *designs* de produtos. Estas rápidas mudanças exigem que os processos de fabricação sejam constantemente redefinidos, buscando redução de custos e aumento de produtividade aliado à qualidade. Desta forma, métodos de monitoramento tem sido estudados e desenvolvidos a fim de identificar alterações, falhas ou desgastes no processo.

Foram analisados um total de 15 artigos relacionados ao monitoramento de processos de fabricação, utilizando os seguintes critérios de seleção:

- foco no monitoramento de um processo de fabricação;
- utilização de sinais para monitoramento;
- utilização da extração de parâmetros de sinais para modelamento;
- utilização de classificadores;

Com base nos critérios listados, as 15 publicações analisadas podem ser apresentadas de acordo com a distribuição cronológica demostrada na Figura 2.1.

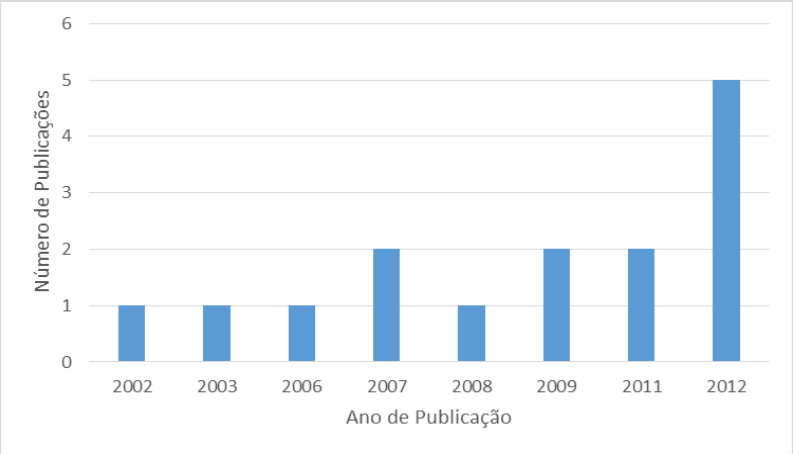


Figura 2.1– Distribuição anual das publicações revisadas.

Fonte: Próprio autor.

Os processos de fabricação específicos de cada trabalho investigado são indicados na Figura 2.2.

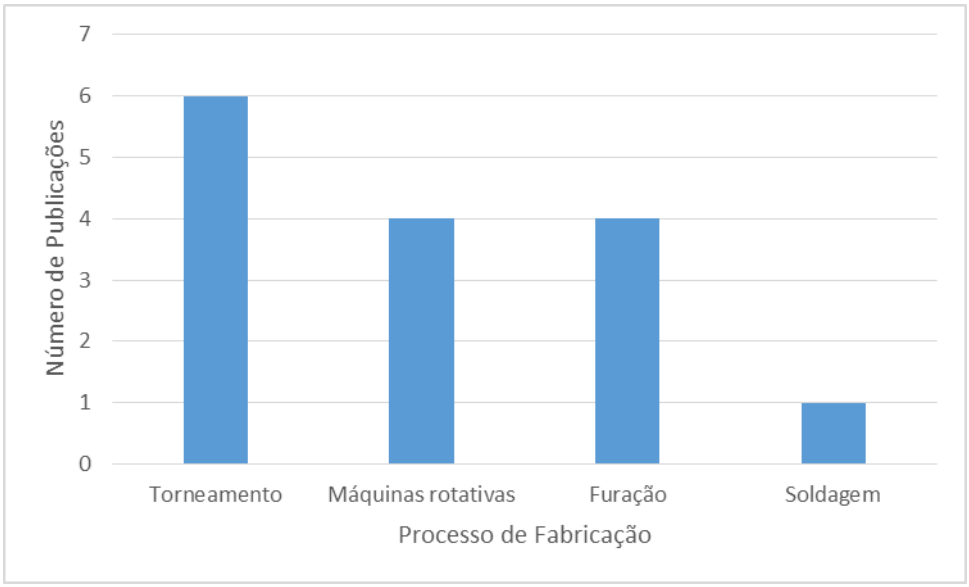


Figura 2.2 – Processo de fabricação estudado nas publicações revisadas.

Fonte: Próprio autor.

Os métodos de monitoramento podem ser classificados como diretos ou indiretos (SICK, 2002). A maioria dos métodos diretos exige que o processo seja interrompido para que sejam efetuadas medidas em alguma parte específica, como por exemplo, o desgaste de material da ferramenta ou a rugosidade da peça usinada. Por esta razão, o grande destaque tem sido os métodos de monitoramento indiretos, por não precisarem parar ou interferir no processo, o que aumenta a eficiência do sistema e permite ajustes em tempo real.

Muitos sinais podem ser utilizados para monitoramento indireto de processos como, por exemplo, os sinais de força de corte (SR E LISTER, 2000), sinais de vibração mecânica (XIQING E CHUANGWEN, 2009), corrente elétrica, potência, energia, temperatura de corte, imagem (KASBAN *et al.*, 2011), entre outros. Dentre os sinais mais utilizados, pode-se destacar os sinais de vibração mecânica, que podem ser classificados de acordo com a faixa de frequência da vibração. Os sinais de vibração, amostrados com frequências na faixa de alguns MHz, são chamados de sinais de emissão acústica (EA) e normalmente são capturados utilizando sensores de vibração (acelerômetros) acoplados a alguma parte do equipamento.

Já os sinais de vibração na faixa de frequência audível (até 20 kHz) são comumente conhecidos como sinais acústicos (sons) e são capturados através de microfones.

A distribuição dos tipos de sinais utilizados nos métodos de monitoramento dos trabalhos analizados é apresentado na Figura 2.3.

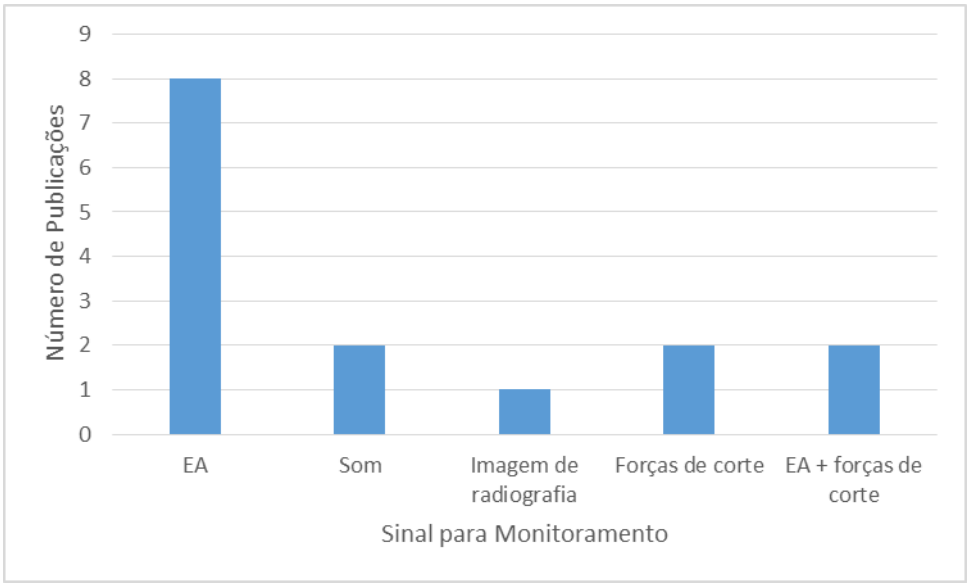


Figura 2.3 – Sinal para monitoramento estudado nas publicações revisadas.

Fonte: Próprio autor.

A grande vantagem da utilização dos sinais de EA é que a faixa de frequência do sinal está acima das vibrações da máquina e dos ruídos do ambiente (LI, 2002). É possível identificar alterações nas características do sinal de emissão acústica, em processos de torneamento, a partir de alterações nas condições de corte e no desgaste da ferramenta (FADARE *et al.*, 2012).

Desta forma, é possível utilizar o sinal de emissão acústica para monitorar alterações nos parâmetros do processo e identificar possíveis falhas ou alterações na qualidade. Devido a esta característica, este tipo de sinal tem sido muito utilizado para o monitoramento de processos de torneamento.

Assim como o sinal de EA, o som também tem sua importância no monitoramento de processos (GRABEC *et al.*, 1998). Apesar de pouco explorados, os sinais acústicos apresentam características semelhantes aos sinas de emissão acústica, além de apresentarem, como grande vantagem, o tipo de sensor (microfone) utilizado para captura do sinal. O principal benefício dos sensores de sinais acústicos está associado com a natureza física do mesmo. Eles são, em geral, fáceis de montar, podendo ser instalados próximos ao ponto de usinagem, com pouca ou nenhuma interferência na máquina, na ferramenta ou peça a ser usinada (RUBIO E TETI, 2002). Apresentam baixo custo comparando-se aos sensores utilizados para aquisição dos sinais

de EA (RAJA, LIM E VENKATASESHAIAH, 2012). Este trabalho utilizará os sinais acústicos para o método de monitoramento devido à vantagem na aquisição apresentada além de ser um sinal pouco explorado no meio científico, conforme apresentado na Figura 2.3.

Associado à escolha do sinal está a técnica de processamento de sinais que contempla a extração de parâmetros e o modelamento estatístico. Diferentes parâmetros foram utilizados em métodos de monitoramento, como por exemplo, média, variância, curtose, etc. (domínio do tempo), transformada de pacotes Wavelet (ZHU, WONG E HONG, 2009), energia, MFCC, etc. (domínio da frequência). Uma distribuição dos parâmetros utilizados nas publicações estudadas é apresentada na Figura 2.4.

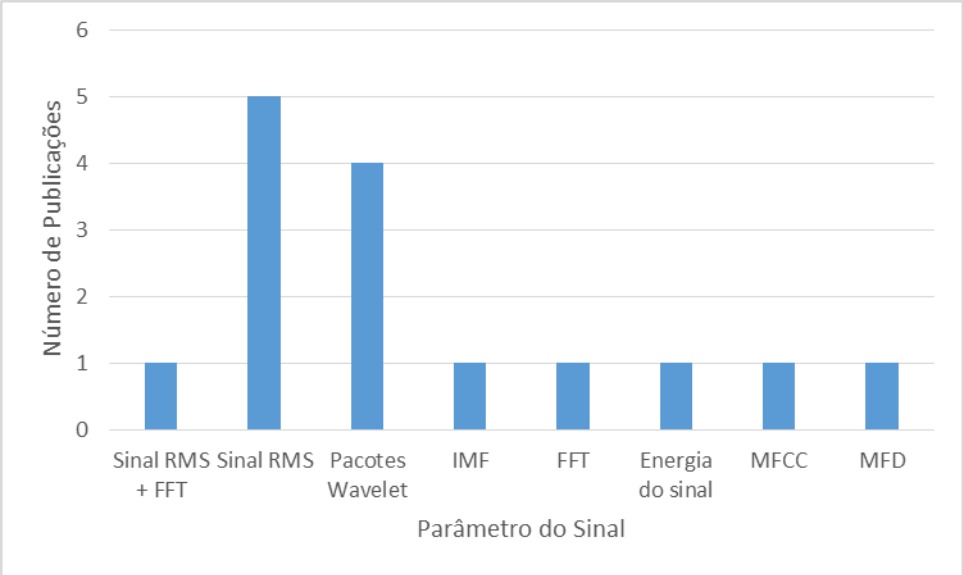


Figura 2.4 – Parâmetro do sinal estudado nas publicações revisadas.

Fonte: Próprio autor.

Estes parâmetros são utilizados para treinar modelos estatísticos que mais tarde são utilizados como classificadores de padrões. Dentre os classificadores que ganham destaque estão os baseados em GMM (MARWALA, MAHOLA E NELWAMONDO, 2006), *Hidden Markov Models* - HMM (BOUTROS E LIANG, 2011; ERTUNC, LOPARO E OCAK, 2001), e *Artificial Neural Networks* - ANN (BIN *et al.*, 2012; HU *et al.*, 2007). Os classificadores utilizados nos trabalhos estudados nesta pesquisa podem ser separados de acordo com a distribuição da Figura 2.5.

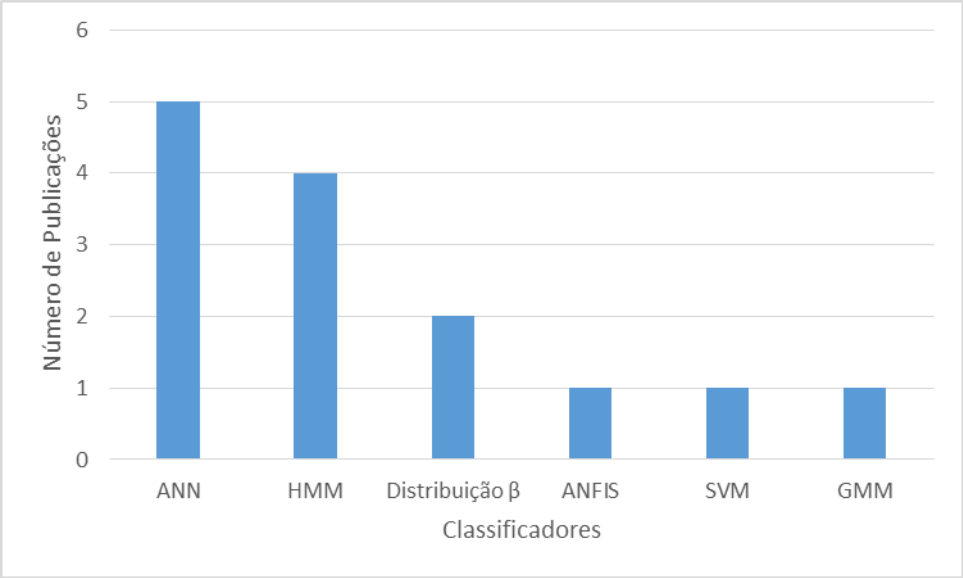


Figura 2.5 – Classificador de padrões estudado nas publicações revisadas.

Fonte: Próprio autor.

Baseando-se na habilidade e sensibilidade dos operadores experientes, que conseguem identificar características das etapas do processo de acordo com o som emitido, a técnica de processamento utilizando o som tende a modelar a forma de ouvir do operador. Desta forma, as técnicas de monitoramento que utilizam o som se assemelham às técnicas de processamento de voz e de sinais de áudio, no que tange a extração de parâmetros e modelamento estatístico dos sinais (ZHU, WONG E HONG, 2009). Estas técnicas utilizam parâmetros como o MFCC (KOOLAGUDI, RASTOGI E RAO, 2012) e o perfil de energia. Para o modelamento estatístico e classificação dos modelos, utiliza-se o GMM (DHANALAKSHMI, PALANIVEL E RAMALINGAM, 2011; KUMARI, NIDHYANANTHAN E G, 2012). Baseando-se nestas técnicas, a proposta é criar um método de monitoramento capaz de identificar e separar grupos de níveis de rugosidade. A utilização do som como sinal para monitoramento, assim como o MFCC como parâmetro para alimentar classificadores GMM é motivado pela ausência de trabalhos que exploram estes conceitos em métodos de monitoramento de processos de fabricação, conforme pode ser visto na Figura 2.4 e na Figura 2.5.

O MFCC é comumente utilizado em outras áreas de pesquisa mas ainda não foi explorado em métodos de monitoramento. Isto faz deste parâmetro uma grande contribuição deste trabalho, ao ser proposto para o monitoramento da rugosidade em processo de torneamento. Foram selecionadas e revisadas 15 publicações que utilizam o MFCC em diversas áreas e que podem ser vistas na distribuição da Figura 2.6.

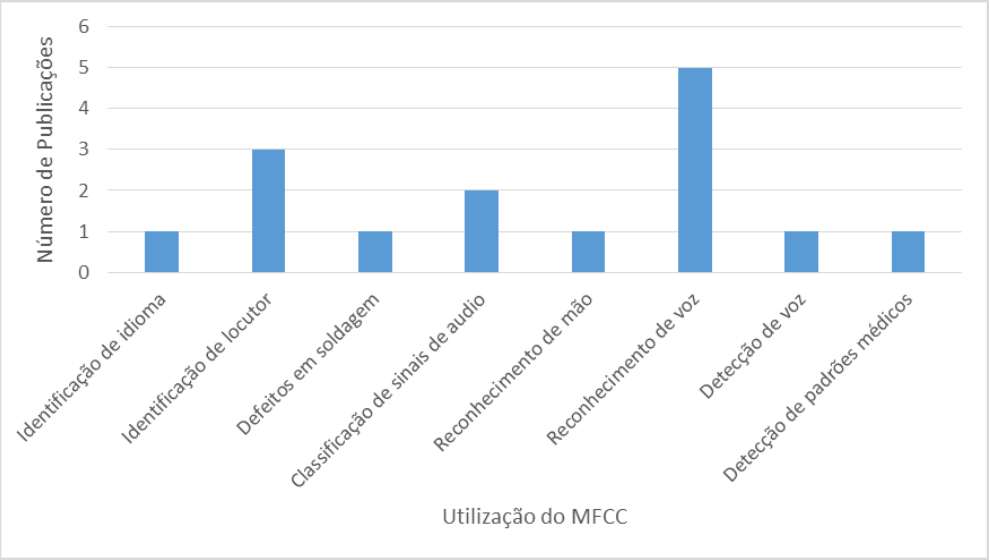


Figura 2.6 – Utilização do MFCC nas publicações revisadas.

Fonte: Próprio autor.

Doze dos quinze trabalhos selecionados utilizam o MFCC para processos de reconhecimento de padrões de voz, dentre os quais reconhecimento de voz (JUNQIN E JUNJUN, 2011; MAYORGA, BESACIER E LAMY, 2003; POLUR E MILLER, 2006; WANG, MIAO E MENG, 2008; YUJIN, PEIHUA E QUN, 2010), reconhecimento de locutor (KAMRUZZAMAN *ET AL.*, 2010; KUMARI, NIDHYANANTHAN E G, 2012; ZUNJING E ZHIGANG, 2005), identificação de idioma (KOOLAGUDI, RASTOGI E RAO, 2012) e classificação de sinais de áudio (DHANALAKSHMI, PALANIVEL E RAMALINGAM, 2009, 2011). Os três trabalhos remanescentes tratam de outros processos como, identificação de defeitos em soldagem (KASBAN *et al.*, 2011), reconhecimento de palma de mão (FAHMY, 2010) e identificação de sons para fins médicos (CHAUHAN *et al.*, 2008).

O processo de monitoramento pode ser dividido em duas etapas: treinamento e teste.

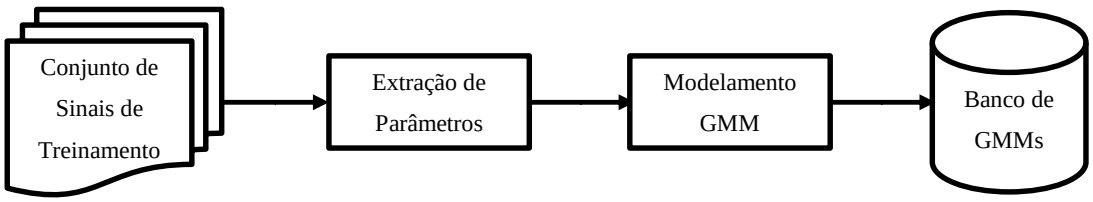


Figura 2.7 – Diagrama das etapas do processo de treinamento.

No processo de treinamento, representado pela Figura 2.7, apresenta-se uma sequência de sinais com uma mesma característica do processo, por exemplo, mesma rugosidade, e cria-se um modelo estatístico baseado no parâmetro extraído de cada sinal do conjunto. Ao fim do processo de treinamento, cria-se um banco com modelos estatísticos onde cada modelo representa a característica que define um determinado conjunto de sinais.

Finalmente, na etapa de teste, representada pela Figura 2.8, ao apresentar um sinal desconhecido ao sistema treinado, o mesmo extrai os parâmetros deste e os compara com os vários modelos existentes no banco, retornando o modelo com maior probabilidade de representar o sinal de entrada.

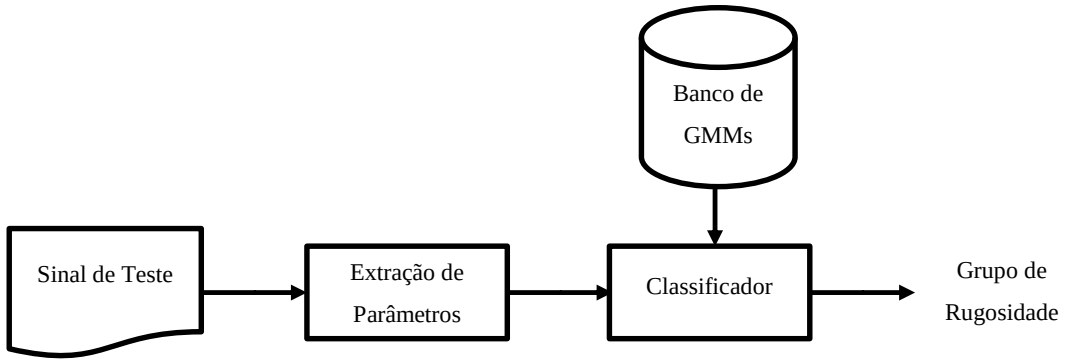


Figura 2.8 - Diagrama das etapas do processo de teste.

Para melhor detalhamento do método de monitoramento proposto, a próxima seção abordará o bloco de extração de parâmetros dos sinais acústicos, mostrando o processo de extração dos coeficientes mel-cepstrais e do perfil de energia.

2.2 Extração de Parâmetros

O sinal amostrado e representado no domínio do tempo, muitas vezes, não apresenta informações relevantes capazes de identificar o fenômeno físico que o gerou. Desta forma faz-se necessário converter o sinal para o domínio da frequência a fim de extrair características que consigam maximizar a separação destes sinais em diferentes classes. A extração de parâmetros de sinais acústicos pode ser dividida em 2 etapas básicas (KOOLAGUDI, RASTOGI E RAO, 2012; PICONE, 1993): o janelamento do sinal e a análise espectral, conforme apresentado na Figura 2.9.

A seguir serão apresentados os detalhes do processo de janelamento do sinal e a necessidade de utilização desta etapa. Posteriormente, a etapa de análise espectral será descrita, apresentando o método de extração do MFCC e do perfil de energia.

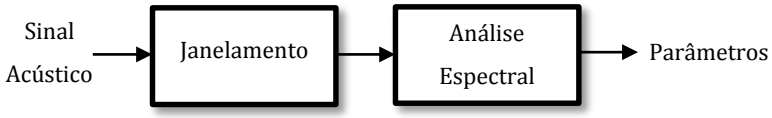


Figura 2.9 - Diagrama em blocos do processo de extração de parâmetros

2.2.1 Janelamento

Como são raras as referências para a análise de sinais acústicos provenientes de processos de usinagem, os tempos envolvidos no processamento serão baseados nos sinais de voz, por apresentarem as mesmas características acústicas. Desta forma, a análise espectral deve ser efetuada em janelas de 20ms deslocadas de 10ms (KOOLAGUDI, RASTOGI E RAO, 2012; PICONE, 1993). Portanto, a cada intervalo de 10ms, uma janela de 20ms do sinal é filtrada por uma janela de Hamming. Esta janela tem por finalidade produzir uma estimativa suavizada da energia, através de regiões onde a mesma muda rapidamente, e aumentar a continuidade entre janelas adjacentes (FAHMY, 2010; PICONE, 1993). Desta forma, tem-se uma superposição das amostras entre duas janelas adjacentes, conforme Figura 2.10.

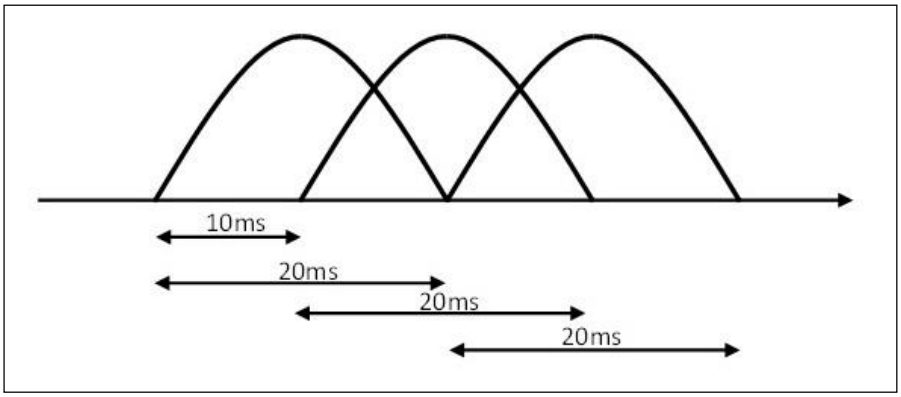


Figura 2.10 - Superposição de janelas

A janela de Hamming pode ser calculada utilizando a seguinte equação (DHANALAKSHMI, PALANIVEL E RAMALINGAM, 2011):

$$\mathrm{w(n)} = 0{,}54 - 0{,}46\cos\left(\frac{2\pi \mathrm{n}}{\mathrm{N}-1}\right),\ 0 \leq \mathrm{n}\ \leq \mathrm{N}-1 \tag{2.1}$$

onde N é o tamanho da janela do sinal . A equação do sinal janelado fica da seguinte forma (FAHMY, 2010):

$$\tilde{x}_l(n) = x(n)w(n),\ 0 \leq n\ \leq N-1 \tag{2.2}$$

onde $x(n)$ é o sinal no domínio do tempo e $w(n)$ é a janela de Hamming.

Depois de multiplicado pela janela de Hamming, o sinal está pronto para a etapa de análise espectral onde é possível extrair parâmetros através da conversão da representação temporal para alguma forma de representação espectral. Nas seções a seguir, serão apresentados os métodos de extração dos coeficientes mel-cepstrais e do perfil de energia, a partir de uma janela do sinal observado.

2.2.2 Coeficientes Mel-Cepstrais (MFCC)

Os coeficientes mel-cepstrais (MFCCs) são uma representação da forma que ouvimos o sinal.

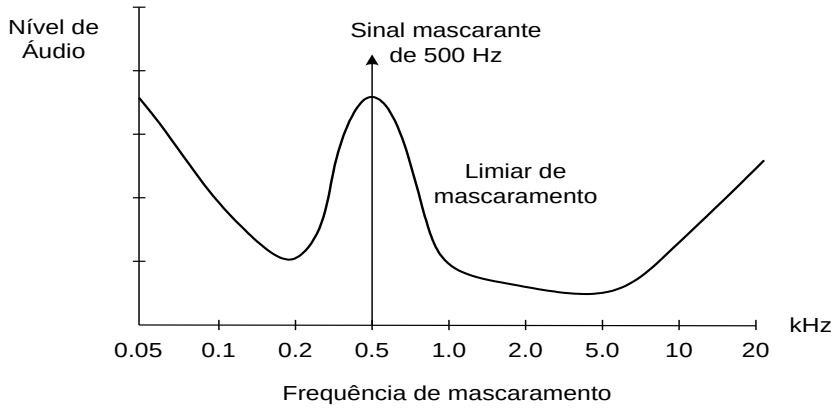


Figura 2.11 - Exemplo de banda crítica

Experimentos em percepção humana mostraram que existe uma banda crítica, com largura entre 10% e 20% da frequencia central do som, onde uma determinada frequência não consegue ser identificada individualmente, ou seja, conforme exemplificado na Figura 2.11, um tom mais alto mascara tons mais baixos de frequencias próximas (PICONE, 1993).

Desta forma, arranja-se linearmente um banco de filtros passa faixa FIR ao longo da escala Mel onde as larguras de faixa de cada filtro são escolhidas para serem iguais às bandas críticas para a frequência central correspondente.

O diagrama do método de extração dos parâmetros Mel, proposto por Mermelstein (1980), está representado na Figura 2.12 e será detalhado na sequência.

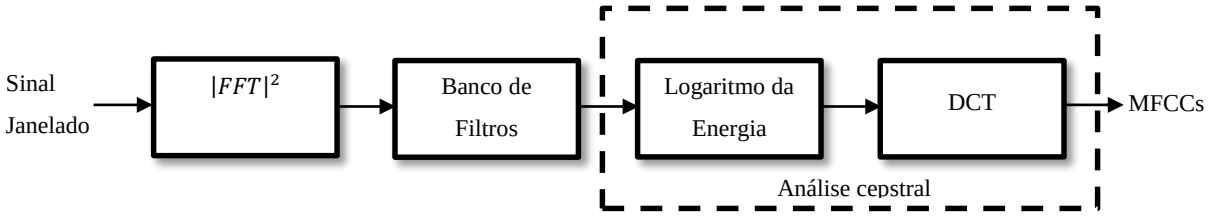


Figura 2.12- Diagrama em blocos do processo de extração de MFCC

Primeiramente, deve-se calcular a energia através do módulo da FFT (F*ast Fourier Transform*) ao quadrado conforme a equação (KOOLAGUDI, RASTOGI E RAO, 2012; KUMARI, NIDHYANANTHAN E G, 2012):

$$S(k) = |X(k)|^2 \tag{2.3}$$

onde $X(k)$ representa a janela do sinal, transladada para o domínio da frequência, através da FFT.

Em seguida filtra-se este sinal com um banco de filtros Mel, também chamado de banco de filtros de Davis-Mermelstein. Este banco de filtros é composto por filtros triangulares com frequências centrais definidas pela seguinte regra (PICONE, 1993):

1. A frequência central varia linearmente até a frequência de 1 kHz;
2. A partir de 1KHz, a frequência varia exponencialmente com um fator de $2^{1/5}$.

A frequência na escala Mel pode ser encontrada através da equação (KASBAN *et al.*, 2011; KOOLAGUDI, RASTOGI E RAO, 2012; PICONE, 1993):

$$f_{mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.4}$$

onde f é a frequência ao qual se deseja transformar para a escala Mel. Desta forma, a Tabela 2.1 a seguir apresenta os valores iniciais das frequências centrais de um banco de filtros Mel obtidos através da Equação 2.4.

Tabela 2.1- Tabela com as frequências centrais para o banco de filtros mel

Indíce dos Filtros	**Freq. Central (Hz)**	**Índice dos Filtros**	**Freq. Central (Hz)**	**Índice dos Filtros**	**Freq. Central (Hz)**
1	100	13	1516	25	8000
2	200	14	1741	26	9190
3	300	15	2000	27	10556
4	400	16	2297	28	12126
5	500	17	2639	29	13929
6	600	18	3031	30	16000
7	700	19	3482	31	18379
8	800	20	4000	32	21112
9	900	21	4595	33	24251
10	1000	22	5278	34	27858
11	1149	23	6063	35	32000
12	1320	24	6964	36	36758

O banco de filtros na escala Mel pode ser implementado a partir de um conjunto de filtros passa-faixa triangulares onde, o *k*-ésimo filtro (*k* corresponde ao índice do filtro) é construído com frequência central $f(k)$, com suas faixas se estendendo da frequência $f(k-1)$ à frequência $f(k+1)$, de modo que $f(k+1)$ não ultrapasse $f_s/2$ (onde f_s representa a frequência de amostragem). A 0 mostra uma representação do banco de filtros (VIGNOLO *et al.*, 2011).

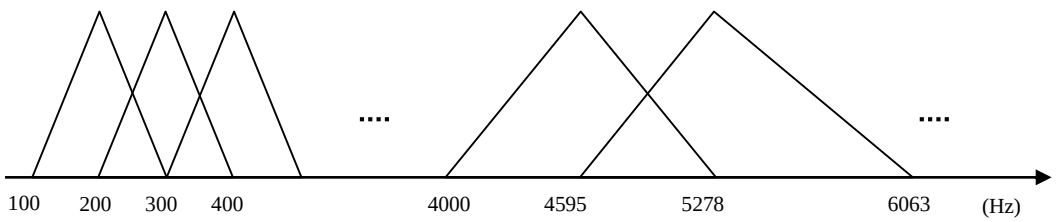


Figura 2.13 - Representação de um banco de filtros triangulares na escala mel

A equação do sinal filtrado fica da seguinte forma (KASBAN *et al.*, 2011; KUMARI, NIDHYANANTHAN E G, 2012):

$$\tilde{s}(l) = \sum_{k=0}^{\frac{N}{2}} S(k) M_l(k), \;\; l = 0,1,\dots,L-1 \tag{2.5}$$

onde N é o comprimento da FFT e L o número de filtros Mel.

Finalmente, aplica-se a *Discrete Cosine Transform* (DCT) ao logaritmo natural do espectro Mel para obter os coeficientes Mel-cepstrais, conforme a equação (DHANALAKSHMI, PALANIVEL E RAMALINGAM, 2011; FAHMY, 2010; PICONE, 1993):

$$c(m) = \sum_{i=0}^{L-1} \ln\big(\tilde{s}(i)\big) \cos\left(\frac{\pi m}{2L}(2i+1)\right), \; m = 0,1,\dots,C-1 \tag{2.6}$$

onde C é o número de coeficientes desejados. Como a maior parte da informação do sinal é representada pelos primeiros coeficientes, normalmente utiliza-se o número de coeficientes entre 12 e 20.

Em seguida será apresentado o método de extração do perfil de energia.

2.2.3 Perfil de Energia.

O perfil de energia é uma representação da distribuição de energia ao longo do espectro de frequência. Portanto, F_K é a frequência a qual abaixo está concentrada uma porcentagem K de energia. O método para extração do perfil de energia pode ser representado pelo diagrama apresentado na Figura 2.14.

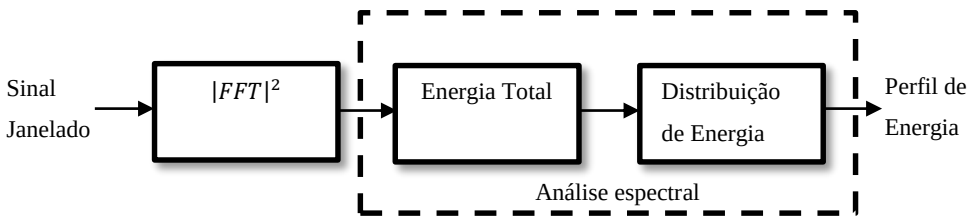


Figura 2.14 - Diagrama em blocos do processo de extração do perfil de energia

Somando-se a energia de todos os pontos da FFT do sinal janelado $X(k)$, obtem-se a energia total da janela, como pode ser visto na equação (ARAÚJO, 2000):

$$E_T = |X(0)|^2 + 2 \times \sum_{k=1}^{N/2} |X(k)|^2 \tag{2.7}$$

onde N é o número de pontos da FFT.

O próximo passo é determinar o número de níveis que representará a distribuição de energia. Quanto maior o número de níveis, maior a resolução do perfil de energia. Supondo-se que o número de níveis seja definido como β, a energia total será dividida em β percentuais de acordo com a equação:

$$R\% = {}^{100\%}/_{(\beta + 1)} \tag{2.8}$$

onde *R* representa a resolução do perfil de energia.

Exemplificando, para β=4, a resolução *R* será de 20% para cada nível do perfil. Em valores de energia, a equação (2.8) pode ser representada da seguinte forma:

$$R = {}^{E_T}/_{(\beta + 1)} \tag{2.9}$$

Partindo-se da resolução calculada em (2.9), inicia-se um processo de somatório dos valores da energia, de cada ponto da FFT do sinal janelado, até atingir o valor de um limiar, representado pelo valor da resolução. Ao atingir este limiar, anota-se o valor da frequência correspondente, F_1. Para cada nível do perfil, atualiza-se o valor do limiar somando-se o valor da resolução R e repete-se o processo obtendo-se $F_2, F_3, \ldots, F_\beta$. Como resultado obtem-se β valores de frequência para cada janela do sinal representando a distribuição percentual de energia conforme Figura 2.15. O algoritmo para o cálculo do perfil de energia está detalhado no Anexo B.

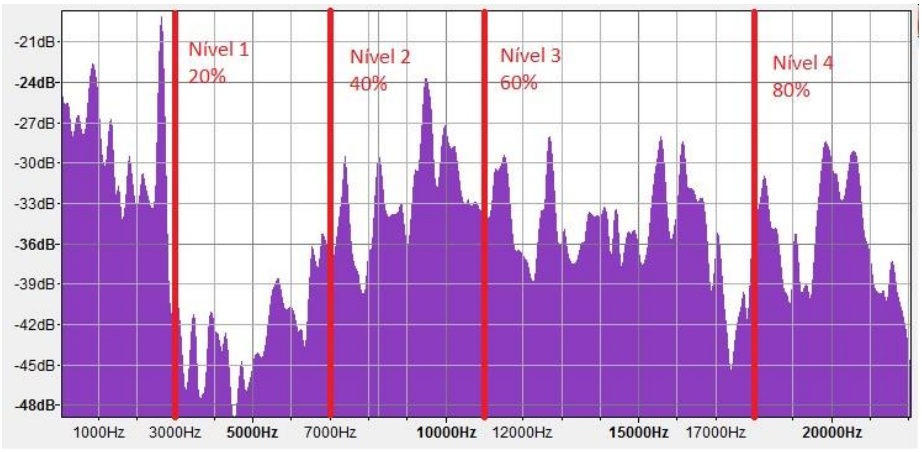


Figura 2.15 - FFT de um sinal janelado com perfil de energia para $\beta = 4$.

Fonte: Próprio autor.

Observando-se o perfil de energia de todas as janelas do sinal, é possível identificar o comportamento da distribuição de energia ao longo do tempo. A Figura 2.16 apresenta dois gráficos, um com o perfil de energia de um sinal capturado durante o processo de usinagem para $\beta = 4$, porém com apenas três níveis plotados (40%, 60% e 80%), e outro com o espectrograma do mesmo sinal para fins comparativos.

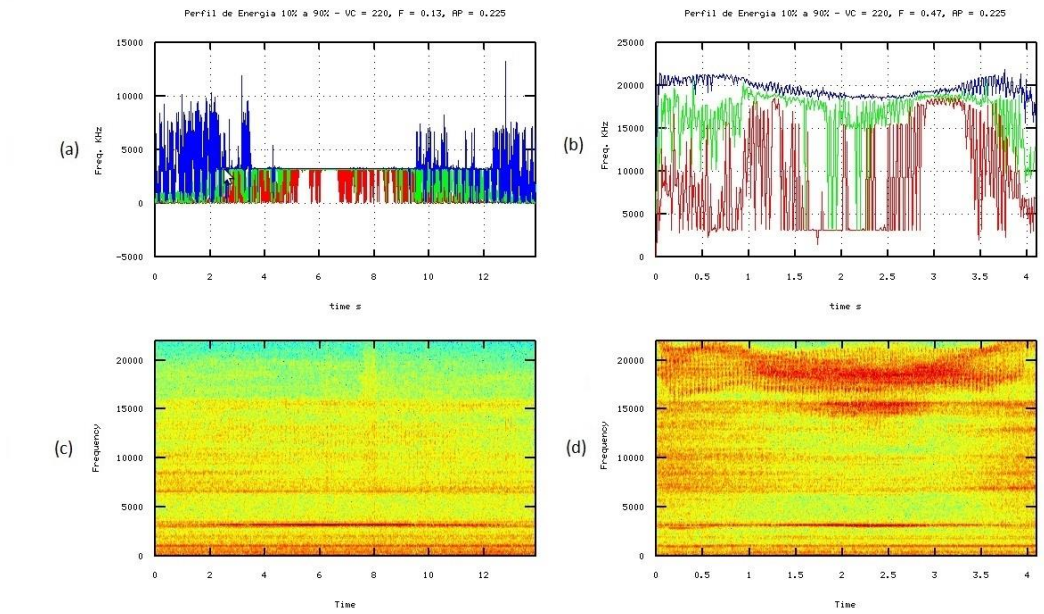


Figura 2.16 - (a) e (b) Perfil de energia 40%, 60% e 80%. (c) e (d) Espectrograma.

Fonte: próprio autor.

Nas Figuras 2.16 (a) e (b), cada linha representa a variação da energia para cada limiar β em relação a frequência, ao longo de todo o tempo analisado. Ao comparar com os espectrogramas das Figuras 2.16 (c) e (d) é possível verificar que a maior concentração de energia do sinal, representada pelos tons mais avermelhados, estão localizados nos lugares próximos ao limiar β de maior valor, no caso representado pela linha azul.

Depois de extrair os parâmetros dos sinais, como o MFCC mostrado na seção 2.2.2 e o perfil de energia na seção 2.2.3, estes são utilizados para a geração de modelos estatísticos que posteriormente são usados para a classificação dos sinais. A próxima seção abordará os tipos de classificadores utilizados para identificação de sinais acústicos baseados em parâmetros de sinais.

2.3 Principais Modelos Utilizados para Classificação de Sinais.

Diferentes técnicas de modelamento são utilizados para identificar e classificar padrões em sinais. Para sinais onde a informação temporal é importante, o modelamento utilizando *Hidden Markov Models* (HMM) apresentam bom desempenho no modelamento tanto de sinais dinâmicos como estáticos (BOUTROS E LIANG, 2011). Por outro lado, quando a informação temporal não é relevante, ganham destaques os classificadores baseados em Redes Neurais Artificiais (*ANN – Artificial Neural Networks*) ou Modelos de Mistura de Gaussianas (*GMM – Gaussian Mixture Models*). Para o GMM, todo o sinal é tratado com uma entidade singular, representado por uma única densidade espectral (BROWN E SMARAGDIS, 2009), ou seja, não apresenta variação nos parâmetros acústicos ao longo do tempo. Desta forma, podemos ver o GMM como um HMM de um único estado.

Analisando sinais capturados durante o processo de torneamento, verificou-se que seus parâmetros aparentemente não se alteravam com o tempo, e portanto, poderiam ser modelados como um processo estacionário. Desta forma, definiu-se que o classificador para este trabalho seria baseado em GMM.

2.4 Modelos de Mistura de Gaussiana (GMM).

Na abordagem utilizando o modelo de mistura de Gaussianas, cada *cluster* formado pelas características do sinal é identificado e modelado por uma função densidade de probabilidade. Dado um vetor $\vec{x}$ com uma sequência D de parâmetros, a densidade de misturas

utilizada para a função de verossimilhança pode ser definida conforme equação (KUMARI, NIDHYANANTHAN E G, 2012; REYNOLDS, QUATIERI E DUNN, 2000):

$$p(\vec{x} \mid \lambda) = \sum_{i=1}^{M} \omega_i p_i(\vec{x}) \tag{2.10}$$

A densidade é uma combinação linear ponderada de M densidades de Gaussianas, representadas por $p_i(\vec{x})$ onde cada uma é parametrizada por um vetor média $D \times 1$, representado por $\vec{\mu}_i$, e uma matriz de covariância $D \times D$, representada por $\sum i$ (DHANALAKSHMI, PALANIVEL E RAMALINGAM, 2011; KUMARI, NIDHYANANTHAN E G, 2012); Para este trabalho foi considerada a utilização de uma matriz de covariância diagonal, ou seja, os elementos de $\vec{x}$ foram considerados independentes.

$$p_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}|\sum i|^{1/2}} exp\left\{-\frac{1}{2}(\vec{x} - \vec{\mu}_i)'(\sum i)^{-1}(\vec{x} - \vec{\mu}_i)\right\} \tag{2.11}$$

onde os pesos das misturas, denotados por w_i, deve satisfazer a condição $\sum_{i=1}^{M} w_i = 1$. Os parâmetros do modelo de densidade podem ser definidos como $\lambda = (w_i, \vec{\mu}_i, \sum i)$, onde $i = 1, \ldots, M$. Desta forma, o modelo de mistura de Gaussianas pode ser representado conforme Figura 2.17.

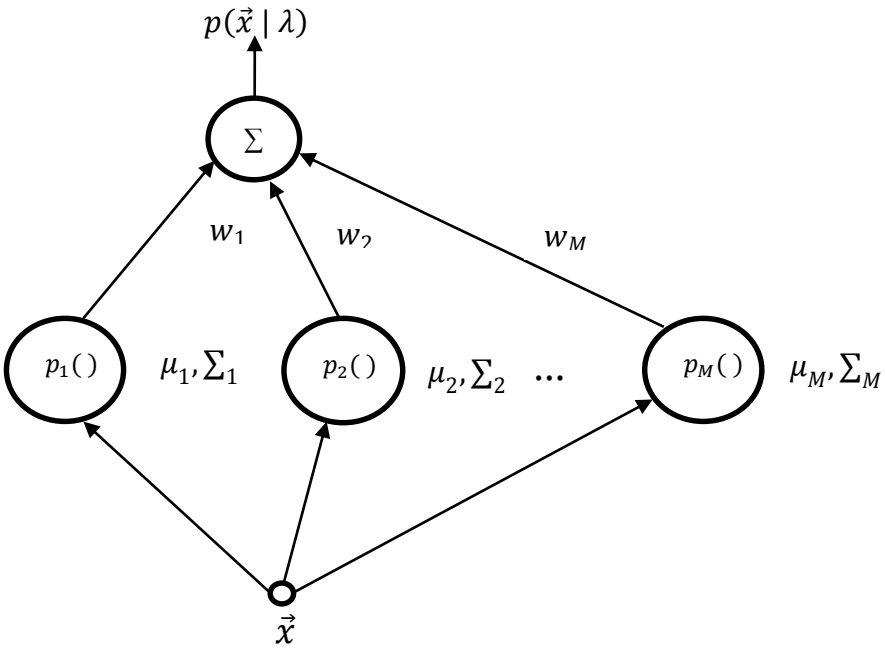


Figura 2.17- Representação de um modelo de misturas com M gaussianas.

O objetivo da estimação dos parâmetros de máxima verossimilhança é encontrar o modelo de parâmetros que maximiza a verossimilhança do GMM. Para uma sequência de T vetores de treinamento $X = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_T)$, a verossimilhança do GMM pode ser escrito como (KUMARI, NIDHYANANTHAN E G, 2012):

$$p(X \mid \lambda) = \prod_{t=1}^{T} p\,(\vec{x}_t \mid \lambda) \tag{2.12}$$

Pelo fato desta expressão ser uma função não linear do parâmetro λ, a maximização direta não pode ser aplicada. Logo, para estimar os parâmetros do modelo de máxima verossimilhança, utiliza-se o algoritmo de iteração *Expectation-Maximization* (EM) o qual ajusta os parâmetros do GMM a cada iteração, aumentando a verossimilhança do modelo estimado para o vetor de observação (REYNOLDS E ROSE, 1995), ou seja, para iterações k e $k+1$, $p(X \mid \lambda^{k+1}) > p(X \mid \lambda^{k})$. A cada iteração, os parâmetros são atualizados segundo as expressões abaixo:

Pesos das misturas:

$$\bar{w}_i = \frac{1}{T}\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda) \tag{2.13}$$

Médias:

$$\vec{\bar{\mu}}_i = \frac{\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda)\vec{x}_t}{\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda)} \tag{2.14}$$

Variâncias:

$$\bar{\sigma}_i^{\,2} = \frac{\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda)\vec{x}_t^{\,2}}{\sum_{t=1}^{T} p(i \mid \vec{x}_t, \lambda)} - \bar{\mu}_i^{\,2} \tag{2.15}$$

onde $\sigma_i^{\,2}, x_t$ e μ_i fazem referência a elementos arbitrários dos vetores $\vec{\sigma}_i^{\,2}, \vec{x}_t$ e $\vec{\mu}_i$ respectivamente.

Logo, a probabilidade após a atualização dos parâmetros fica:

$$p(i \mid \vec{x}_t, \lambda) = \frac{w_i p_i(\vec{x}_t)}{\sum_{k=1}^{M} w_k p_k(\vec{x}_t)} \tag{2.16}$$

Dois pontos críticos no treinamento de modelos de mistura de gaussianas são a escolha do número M de gaussianas na mistura e a inicialização dos parâmetros do modelo para iniciar o algoritmo EM. O método utilizado para a inicialização dos parâmetros neste trabalho foi o Segmental *K-Means* (LINDE, BUZO AND GRAY, 1980).

2.5 Identificação de Padrões de Sinais Modelados por GMM

Segundo Reynolds e Rose (1995), para a identificação de padrões de sinais, um conjunto $\mathrm{S} = \{1, 2, \ldots, \mathrm{S}\}$ de padrões de sinais é representado por um conjunto de GMM denominados $\lambda_1, \lambda_2, \ldots, \lambda_\mathrm{S}$. O objetivo é identificar o modelo que possui a maior probabilidade para uma dada sequência de observação. Formalizando obtém-se a seguinte equação:

$$\hat{S} = \arg \max_{1 \le k \le S} Pr(\lambda_\mathrm{k} \mid X) = \arg \max_{1 \le k \le S} \frac{p(X \mid \lambda_\mathrm{k}) Pr(\lambda_\mathrm{k})}{p(X)} \tag{2.17}$$

onde $Pr(\lambda_\mathrm{k})$ é a probabilidade de ser o grupo λ_k e $p(X)$ é a probabilidade de ocorrer a sequência de observação X a ser identificada. A segunda equação é obtida através da regra Bayes.

Assumindo-se que os grupos de sinais são equiprováveis, ou seja, $Pr(\lambda_\mathrm{k}) = 1/S$, e que $p(x)$ é o mesmo para todos os modelos, a equação pode ser simplificada para:

$$\hat{S} = \arg \max_{1 \le k \le S} p(X \mid \lambda_\mathrm{k}) \tag{2.18}$$

O produtório de probabilidades, que são números menores que 1, vai diminuindo cada vez mais, fazendo com que haja problemas numéricos no cálculo. Portanto, baseando-se na independência entre as observações e utilizando o logaritmo, a equação (2.12) do cálculo identificação de padrões de sinal fica da seguinte forma:

$$\hat{S} = \arg \max_{1 \le k \le S} \sum_{t=1}^{T} \log p(\vec{x}_t \mid \lambda_\mathrm{k}) \tag{2.19}$$

onde $p(\vec{x}_t \mid \lambda_\mathrm{k})$ é calculado por (2.10).

2.6 Metodologia de Superfície de Resposta

Os dados necessários para a pesquisa serão coletados através de um arranjo experimental do tipo CCD para os parâmetros de usinagem do aço ABNT 52100 melhor discutidos no item 3.3. Como serão necessárias muitas réplicas, do CCD original serão selecionados apenas 10 pontos de acordo com o critério de optimalidade D (*D-optmality*) (MONTGOMERY E JOHNSON, 2009).

De acordo com Myers e Montgomery (1995), a Metodologia de Superfície de Resposta (MSR) é uma coleção de técnicas matemáticas e estatísticas que são utilizadas para modelar e analisar problemas nos quais a resposta de interesse é influenciada por muitas variáveis e nos quais a resposta deva alcançar um valor ótimo, e nos quais a forma de relacionamento entre a variável de resposta e as variáveis independentes seja desconhecida. Portanto, o primeiro passo dentro da metodologia MSR é encontrar uma aproximação razoável para o verdadeiro relacionamento entre y e o conjunto de variáveis. Usualmente, emprega-se um polinômio de baixa ordem.

Supondo que a resposta esperada E(Y) seja função de k variáveis preditoras codificadas x1, x2,..., xk, o relacionamento entre y e as variáveis preditoras pode ser expresso segundo uma expansão em Série de Taylor (BOX E DRAPER, 1987), tal que:

$$E(Y) = \eta = \eta_0 + \sum_{i=1}^{k}\left[\frac{\partial \eta}{\partial x_i}\right]_0 x_i + \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\left[\frac{\partial^2 \eta}{\partial x_i \partial x_j}\right] + \ldots \tag{2.20}$$

onde o subscrito zero indica a avaliação na origem.

Se os termos de alta ordem forem ignorados, a expansão produzirá:

$$\eta = \beta_0 + \sum_{i=1}^{k}\beta_i x_i \tag{2.21}$$

Se, em adição, forem mantidos os termos de segunda ordem, a aproximação se tornará:

$$y = \beta_0 + \sum_{i=1}^{k}\beta_i x_i + \sum_{i<j}\sum\beta_{ij} x_i x_i + \varepsilon \tag{2.22}$$

Se houver curvatura no processo, então um polinômio de ordem mais alta deve ser utilizado, tal como um modelo de segunda ordem, por exemplo.

$$y = \beta_0 + \sum_{i=1}^{k}\beta_i x_i + \sum_{i=1}^{k}\beta_{ii} x_i^2 + \sum_{i<j}\sum\beta_{ij} x_i x_{j+}\varepsilon \tag{2.23}$$

Segundo Box e Draper (1987) quase todos os problemas de superfície de resposta utilizam um desses dois modelos. Contudo, é improvável que um modelo polinomial seja uma aproximação razoável do modelo real em todo o espaço experimental, mas pelo menos para uma determinada região, ele funcionará muito bem (MONTGOMERY, 2001). Para estimar os parâmetros (b) do modelo polinomial emprega-se o método dos mínimos quadrados, que também pode ser escrito em forma matricial.

A análise de uma superfície de resposta é feita em termos de uma superfície ajustada. Se tal superfície é adequada, sua análise será aproximadamente equivalente a análise da superfície real. Evidentemente, usando-se um arranjo experimental adequado para coletar os dados de y, os parâmetros do modelo serão estimados eficazmente. Geralmente, quando se está em um ponto da região experimental distante do ponto de ótimo, a curvatura do sistema é bem pequena, de onde decorre que um modelo de primeira ordem seja adequado para modelar a resposta. O objetivo experimental deve ser, portanto, caminhar em direção à região de ótimo. Uma vez encontrada tal região, um modelo quadrático deve ser utilizado.

Sob algumas circunstâncias, um modelo envolvendo apenas efeitos principais e algumas interações pode ser apropriado para descrever a superfície de resposta, principalmente quando a análise dos resultados revelar a inexistência de curvatura. Em outras circunstâncias, a completa descrição do comportamento do processo poderá requerer um modelo quadrático ou cúbico. Se os limites para os fatores forem definidos apropriadamente, será improvável a necessidade de modelos de terceira ordem.

Na maioria dos experimentos, assume-se inicialmente que o modelo linear é adequado; entretanto, para se confirmar se existe ou não falta de ajuste, deve-se utilizar pontos experimentais extras denominados de Pontos Centrais. De acordo com Box et al (1978), tal procedimento consiste em adicionar ao fatorial completo, pontos que sejam intermediários aos níveis dos fatores. Serão adicionados tantos pontos centrais quantos forem os K fatores do experimento. Os pontos centrais são utilizados para avaliar a existência de curvatura. Se a resposta média obtida com os nf pontos fatoriais for significativamente diferente da média formada pelos nc pontos centrais, então, pode-se caracterizar uma região de curvatura significativa. Segundo Montgomery (2001), a soma de quadrados para curvatura é dada por:

$$SS_{QP} = \frac{(nf\ x\ nc)(\bar{y}_f - \bar{y}_c)^2}{nf + nc} \tag{2.24}$$

O Erro médio quadrático e o teste de hipótese são respectivamente:

$$MS_e = \frac{SS_e}{nc-1} = \frac{\sum_{i=1}^{nc}(y_i - \bar{y})^2}{nc-1} \tag{2.25}$$

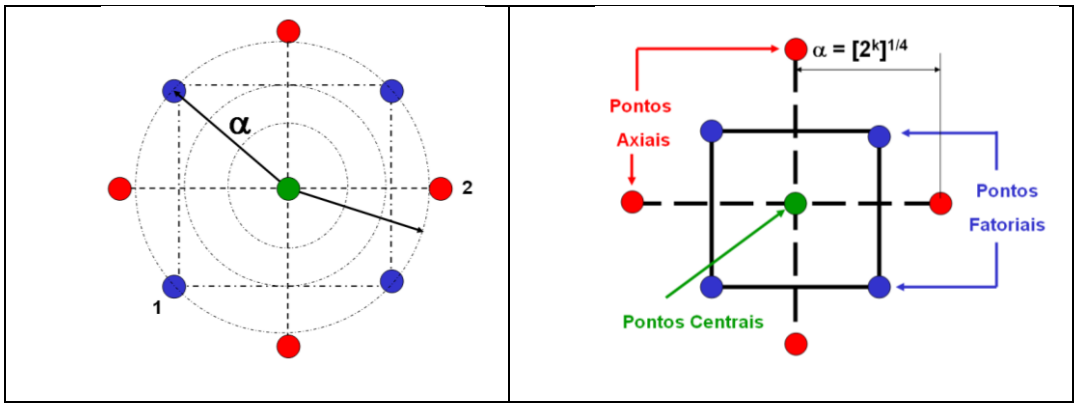
$$H_0 : \sum_{j=1}^{k} \beta_{jj} = 0 \text{ e } H_1 : \beta_{jj} \neq 0 \tag{2.26}$$

Se SS_{qp} for muito maior do que MS_e, haverá evidência suficiente para rejeitar a hipótese nula de que os coeficientes quadráticos são nulos.

2.6.1 Arranjos Experimentais para Superfície de Resposta

Existem dois tipos principais de arranjos experimentais para a coleta de dados na Metodologia de Superfície de Resposta: o arranjo central composto e o Box-Behnken.

O arranjo central composto (Box-Wilson Central Composite Design - CCD) é uma matriz formada por três grupos distintos de elementos experimentais: um fatorial completo ou fracionado; um conjunto de pontos centrais e, adicionalmente, um grupo de níveis extras denominados Pontos Axiais. Se a distância entre o centro do arranjo e ponto fatorial (+1;-1) for aproximadamente 1 (em módulo), a distância do centro ao ponto axial será maior que a unidade. Esta distância, comumente representada por α, depende de certas propriedades desejadas para o experimento e do número de fatores envolvidos (MONTGOMERY, 2001), tal como ilustra a Figura 2.18. O número de pontos axiais em um CCD é igual ao dobro do número de fatores e representam seus valores extremos. Em função de sua localização, podem ser circunscritos, inscritos ou de face centrada.



(a) (b)

Figura 2.18 - (a) Rotacionalidade, (b) Arranjo central composto para 2 fatores.

O Circunscrito (CCC) é o CCD original. Nele, os pontos axiais estão a uma distância α do centro, baseado nas propriedades desejadas do projeto. Este arranjo requer 5 níveis para cada fator.

O Arranjo inscrito (CCI) é adequado às situações nas quais os limites especificados não podem ser extrapolados, quer por medida de segurança, quer por incapacidade física de realização. Neste caso, o CCI utiliza os níveis dos fatores como pontos axiais e cria um fatorial completo ou fracionado dentro desses limites. Um CCI requer 5 níveis.

O arranjo de Face Centrada (CCF) caracteriza-se por dispor os pontos axiais sobre o centro de cada face do espaço fatorial, ou seja, α = +1 ou –1. Requer 3 níveis para cada fator.

Nos arranjos centrais compostos, a variância predita é constante ao longo da esfera de raio α, ou seja, $V\left[\hat{y}(x)\right]_{(1)} = V\left[\hat{y}(x)\right]_{(2)} = ... = V\left[\hat{y}(x)\right]_{(i)}$ e é definida como:

$$V\left[\hat{y}(x)\right] = \sigma^2 x^T \left(X^T X\right)^{-1} x \tag{2.27}$$

Esta propriedade denomina-se rotacionalidade. Segundo Box e Draper (1987), para manter a rotacionalidade de um arranjo, o valor de α depende do número de experimentos (k) da porção fatorial do CCD, tal que:

$$\alpha=(2^k)^{1/4} \tag{2.28}$$

Quando for necessário se empregar um arranjo em blocos, α é dado por:

$$\alpha = \left[k\left(1+\frac{n_{s0}}{n_s}\right) \Big/ \left(1+\frac{n_{c0}}{n_c}\right)\right]^{\frac{1}{2}} \tag{2.29}$$

Na equação (2.29), n_{s0} é o número de pontos centrais presentes na porção axial do arranjo; n_s é a quantidade restante de pontos da porção axial; n_{c0} é o número de pontos centrais da porção cúbica do arranjo (fatorial completo) e n_c é a quantidade restante de pontos da porção cúbica (BOX E DRAPER, 1987).

Um arranjo do tipo Box-Behnken (BOX E BEHNKEN, 1960) é um modelo quadrático independente que não contém fatoriais completos ou fracionários embutidos. Neste tipo de arranjo, as combinações de fatores são os pontos médios das arestas das faces do espaço experimental Ω, além dos pontos centrais, como pode ser visto na Figura 2.19. Estes arranjos são rotacionáveis e requerem 3 níveis para cada fator.

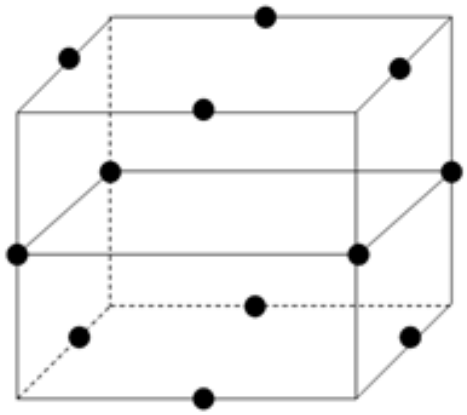


Figura 2.19- Arranjo do tipo Box-Behnken para três fatores.

2.6.2 Critério de Optimalidade D

Como foi descrito no item 2.6.1, um arranjo de superfície de resposta para k=3 fatores, envolveria 8 pontos fatoriais, 6 pontos axiais e 5 pontos centrais, totalizando 19 combinações experimentais. Como para cada combinação experimental são necessárias várias réplicas, a adoção do CCD completo inviabilizaria a condução do estudo, uma vez que as réplicas são importantes para obter consistência na geração dos padrões. Assim neste trabalho, optou-se pela adoção de uma fração do CCD completo escolhida com base no critério de optimalidade D. Este é justamente o caso do processo de torneamento que será estudado neste trabalho uma vez que será avaliada a influência de 3 parâmetros de corte (velocidade de corte, avanço e profundidade de corte) sobre a rugosidade da peça. Como para cada combinação destes parâmetros tenciona-se a obtenção de 15 sinais acústicos (15 réplicas do CCD), isto totalizaria 285 experimentos, quantidade esta proibitiva em relação aos recursos disponíveis para este trabalho.

De acordo com Montgomery (2001), um modelo pode ser considerado *D-optimal* se:

$$|(X'X)^{-1}| \tag{2.30}$$

for minimizado. Desta forma, um modelo *D-optimal* minimiza o volume do intervalo de confiança do vetor de coeficientes de regressão. A equação que mede o desempenho relativo entre o modelo 1 e o modelo 2, baseado no critério de optimalidade D é dado por:

$$D_e = \left(\frac{\left| (X_2' X_2)^{-1} \right|}{\left| (X_1' X_1)^{-1} \right|} \right)^{1/p} \tag{2.31}$$

onde X_1 e X_2 são as matrizes X para os dois modelos e p é o número de parâmetros.

Baseando-se neste critério, diminuiu-se o número de 19 para 10 combinações experimentais, viabilizando assim o trabalho.

3. Material e Método

3.1 Introdução

O método de pesquisa utilizado neste trabalho é o experimental e o processo escolhido foi o de torneamento do aço ABNT 52100, em virtude de sua grande aplicação no mercado industrial por se tratar de um material com alta dureza. Este tipo de material traz benefícios ao processo como redução do tempo de corte, baixa rugosidade da peça, aumento da remoção de material, precisão dimensional e usinagem sem fluido de corte (PAIVA *et al.*, 2012).

O método de monitoramento proposto é baseado em GMM para classificação de padrões, e utiliza o perfil de energia e os coeficientes mel-cepstrais como parâmetros do sinal para modelamento.

3.2 Máquinas, Materiais, Ferramentas e Instrumentos de Medição

No processo de torneamento foi utilizado o torno CNC Nardini Logic 175, com velocidade de rotação máxima de 4.000 rpm e potência de corte de 5,5 kW.

As peças de utilizadas no processo de usinagem possuíam dimensões de Ø 49 mm × 50 mm. Todas as peças foram previamente temperadas e revenidas resultando em uma dureza entre 49 e 52 HRC, até uma profundidade de três milímetros abaixo da superfície.

As ferramentas de corte utilizadas são de cerâmica mista (Al_2O_3+TiC), com geometria alisadora código ISO CNGA 120408 S01525WH revestida com uma camada de nitreto de titânio (TiN). Diferentes ferramentas foram utilizadas a fim de realizar todos os passos com ferramentas não desgastadas.

Os parâmetros de rugosidade foram medidos com um rugosímetro portátil Mitutoyo modelo Surftest SJ-201P.

O som foi gravado utilizando o software Audacity®, com uma frequência de amostragem de 44,1 kHz com resolução de 16 bits.

3.3 Base de Dados

Para treinar os modelos GMM a partir dos parâmetros dos sinais acústicos, primeiramente deve-se montar uma base de dados com gravações de áudio de processos de torneamento e medidas dos diversos parâmetros de rugosidade resultante deste processo.

3.3.1 Medidas de Rugosidade

Para realizar o experimento, elaborou-se um conjunto sequencial de corridas experimentais usando um arranjo composto central (CCD) construído de acordo com Paiva (2012).

Tabela 3.1- CCD para o torneamento do aço ABNT 52100 endurecido.

Vc	f	ap	Ra	Ry	Rz	Rq	Rt
200,000	0,100	0,100	0,1427	1,0502	0,7536	0,1823	1,4384
240,000	0,100	0,100	0,2544	1,9037	1,2891	0,3275	2,7013
200,000	0,200	0,100	0,4397	2,3916	1,7633	0,5263	3,0893
240,000	0,200	0,100	0,4389	2,3247	1,7421	0,5305	4,2837
200,000	0,100	0,200	0,1887	1,3606	1,0027	0,2359	1,8031
240,000	0,100	0,200	0,1850	1,4213	0,9651	0,2417	2,1004
240,000	0,200	0,200	0,5215	2,8989	2,2361	0,6280	3,4129
186,364	0,150	0,150	0,2716	1,8717	1,2522	0,3344	2,7094
220,000	0,234	0,150	0,5062	2,6245	2,0596	0,5971	3,0953
220,000	0,150	0,234	0,1626	1,1710	0,8147	0,2029	1,5570
220,000	0,150	0,150	0,1800	1,2117	0,7833	0,2233	1,6792
220,000	0,150	0,150	0,1767	1,1700	0,7633	0,2183	1,6083
220,000	0,150	0,150	0,1775	1,1900	0,7683	0,2192	1,6333
220,000	0,150	0,150	0,1717	1,3408	0,8458	0,2225	2,0350
220,000	0,150	0,150	0,1717	1,3183	0,8425	0,2217	1,9508
220,000	0,150	0,150	0,1700	1,3025	0,8383	0,2192	1,9783
220,000	0,150	0,150	0,1475	1,0392	0,7533	0,1842	1,3242
220,000	0,150	0,150	0,1475	1,1058	0,7633	0,1858	1,6250
220,000	0,150	0,150	0,1467	1,0467	0,7542	0,1817	1,3675
220,000	0,150	0,150	0,1792	1,3583	0,9492	0,2283	2,0575
220,000	0,150	0,150	0,1758	1,3208	0,9383	0,2233	1,9875
220,000	0,150	0,150	0,1758	1,3217	0,9283	0,2225	2,0100

Devido ao número de ferramentas disponíveis para o experimento, 10 conjuntos de parâmetros de usinagem foram selecionados da Tabela 3.1 usando o critério de *D-optimal design* (MONTGOMERY E JOHNSON, 2009), resultando na Tabela 3.2.

Tabela 3.2- Conjunto de parâmetros de corte para experimento.

Conjunto	V_c (m/min)	f (mm/v)	a_p (mm)
Ra1	186	0,15	0,15
Ra2	220	0,23	0,15
Ra3	220	0,15	0,23
Ra4	240	0,10	0,10
Ra5	240	0,10	0,20
Ra6	240	0,15	0,15
Ra7	240	0,20	0,20
Ra8	200	0,10	0,10
Ra9	200	0,20	0,10
Ra10	200	0,10	0,20

Cada conjunto de usinagem foi executado 15 vezes para formar uma base de dados de treinamento e teste a fim de garantir a confiabilidade estatística do método de monitoramento.

A cada passe, foram medidos 5 parâmetros de rugosidades (rugosidade média aritmética R_a, rugosidade máxima R_y, rugosidade média quadrática R_q, rugosidade média R_z e rugosidade máxima pico / vale R_t) além de gravar o som emitido durante o processo. A rugosidade foi medida quatro vezes em cada ponto (A, B e C) conforme ilustrado na Figura 3.1.

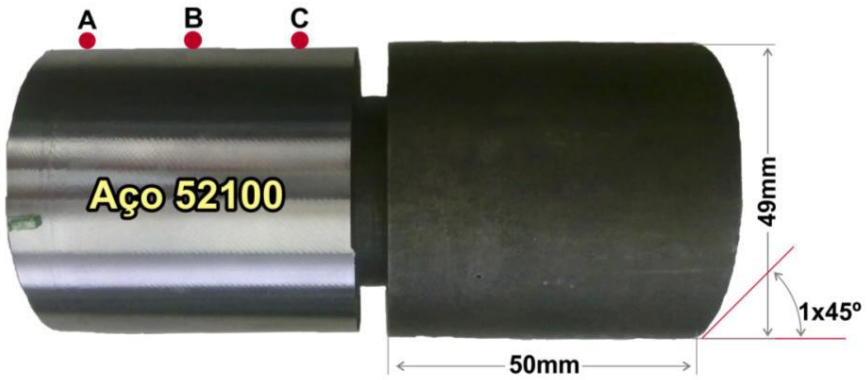


Figura 3.1 - Corpo de prova do aço ABNT 52100.

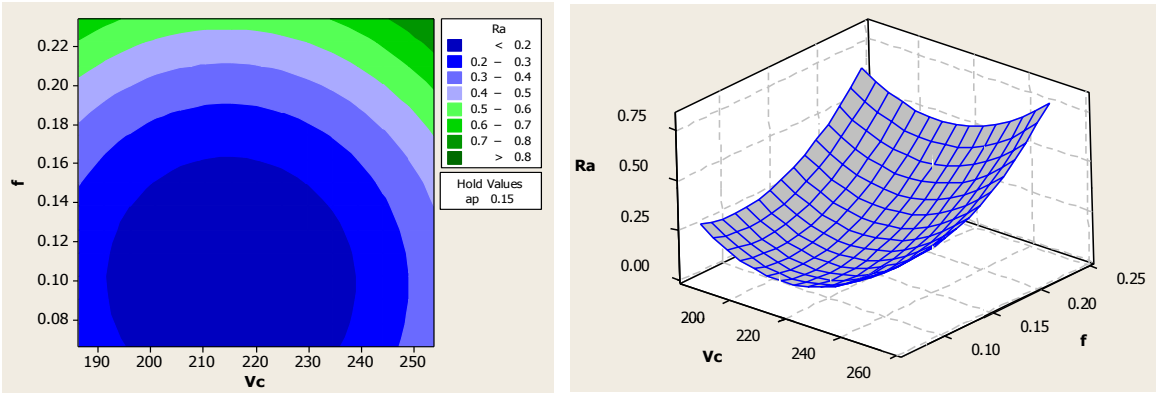
A tabela com as rugosidades medidas pode ser verificada no Anexo A. Os modelos quadráticos completos para as rugosidades do aço ABNT 52100 podem se visualizados na Tabela 3.3. Depois de realizadas as medições, foram calculadas as médias e variâncias das rugosidades medidas.

Tabela 3.3- Modelos Quadráticos Completos para as rugosidades do aço ABNT 52100.

Modelo	Ra	Ry	Rz	Rq	Rt
Constante	**0.1679**[1]	**1.2248**	**0.8254**	**0.2120**	**1.7701**
x_1	**0.0365**	**0.1991**	**0.1494**	**0.0475**	**0.4784**
x_2	**0.1207**	**0.5041**	**0.4061**	**0.1376**	**0.6354**
x_3	-0.0055	0.0205	0.0217	-0.0072	**-0.2320**
x_1^2	**0.0695**	**0.4065**	**0.2883**	**0.0849**	**0.6477**
x_2^2	**0.0590**	**0.2545**	**0.2434**	**0.0677**	0.1219
x_3^2	0.0125	0.0282	0.0319	0.0144	0.0938
x_1x_2	0.0029	-0.0648	-0.0040	0.0018	0.0698
x_1x_3	0.0022	-0.0329	-0.0081	0.0022	-0.1545
x_2x_3	0.0070	0.0988	0.0693	0.0087	-0.1543
$R^2adj._{completo}$	91.87%	85.92%	87.61%	91.13%	87.34%
$R^2adj._{reduzido}$	93.61%	88.73%	90.08%	93.09%	84.28%

[1] – Valores em negrito representam os termos estatisticamente significativos (P-Value < 5%).

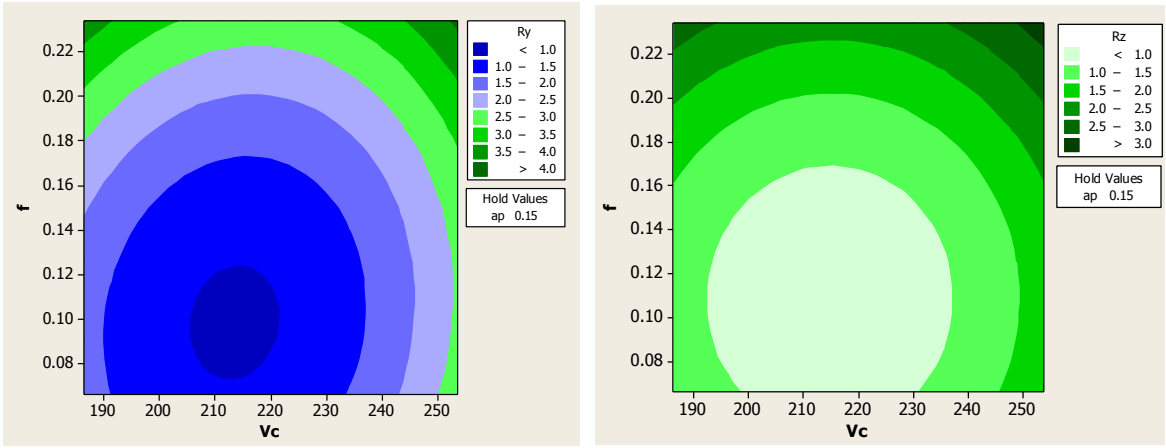
A superfície de resposta e o gráfico de contorno para a rugosidade média Ra é representada pela Figura 3.2. Estes gráficos apresentam a relação entre os parâmetros de corte (velocidade de corte, avanço e profundidade de corte) com a rugosidade resultante. De acordo com as figuras, existe um conjunto de parâmetros que minimiza o valor da rugosidade Ra.



(a) Gráfico de contorno de Ra. (b) Superfície de Resposta para Ra.

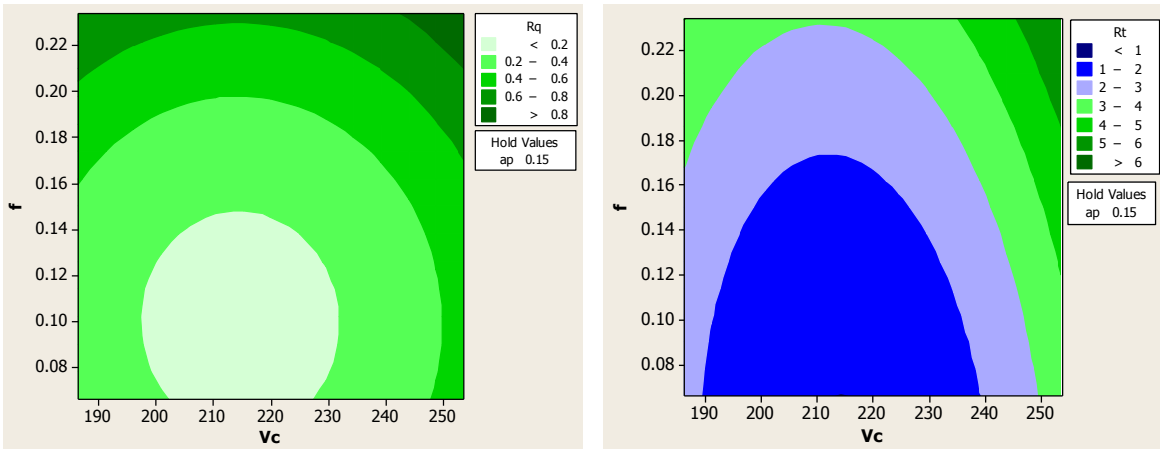
Figura 3.2- Gráficos de contorno e superfície de resposta para a rugosidade Ra.

A Figura 3.3 e a Figura 3.4 apresentam os gráficos de contorno para as demais medidas de rugosidade (Ry, Rz, Rq e Rt). É importante observar os baixos valores para todos os parâmetros de rugosidade obtidos neste trabalho.



(a) Gráfico de contorno de Ry. (b) Gráfico de contorno de Rz.

Figura 3.3 - Gráficos de contorno para a rugosidades Ry e Rz.



(a) Gráfico de contorno de Rq. (b) Gráfico de contorno de Rt.

Figura 3.4 - Gráficos de contorno para as rugosidades Rq e Rt.

3.3.2 Gravação de Áudio

Para a gravação do áudio, foi utilizado o esquema conforme ilustrado na Figura 3.5 (FADARE *et al.*, 2012). No lugar do acelerômetro (sensor de emissão acústica), foi utilizado um microfone acoplado a um computador.

Foram gerados um total de 15 gravações por conjunto de parâmetros de corte totalizando 150 arquivos de áudio para análise.

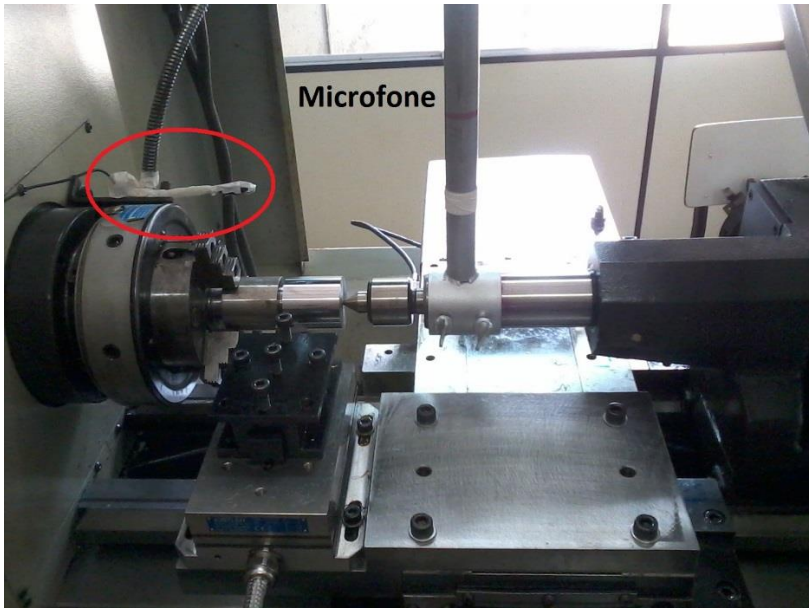


Figura 3.5 - Esquema de ligação para gravação de áudio.

A Figura 3.6 apresenta alguns sinais acústicos no domínio do tempo, gravados durante o experimento, apresentados pelo software Audacity®.

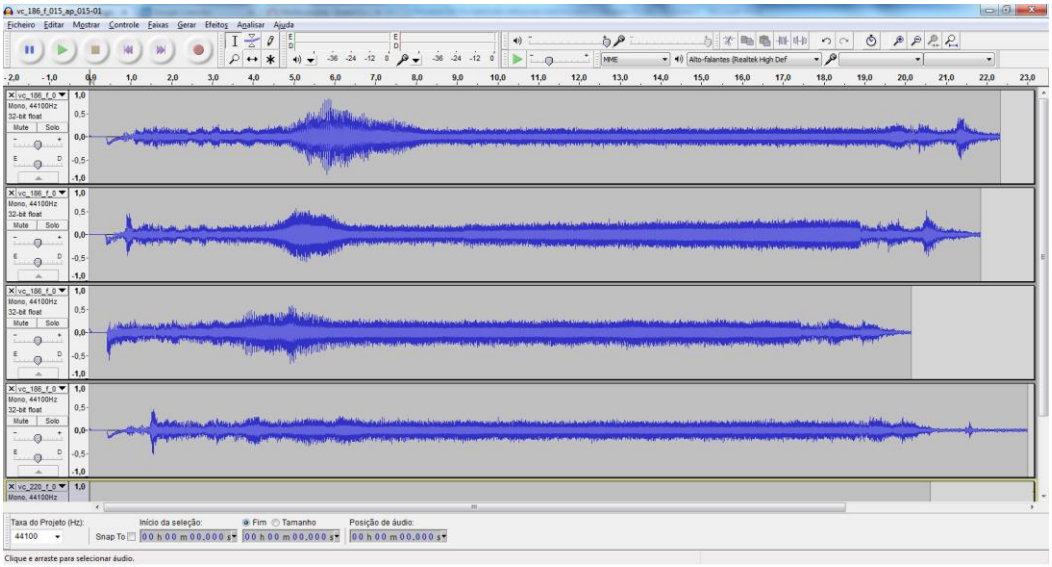


Figura 3.6 - Exemplo de sinais de áudio gravados durante processo.

Para cada arquivo de áudio, foram extraídos intervalos de tempo menores dos sinais, em torno de 7 a 10 segundos, que correspondiam à parte estacionária do sinal, conforme Figura 3.7.

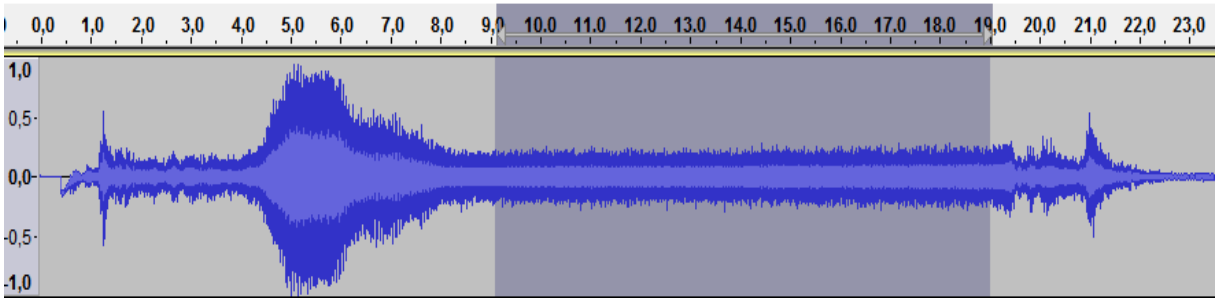


Figura 3.7- Exemplo de sinais de áudio com seleção de intervalo de 10 s.

Com a base de dados formada, a próxima etapa é extrair os parâmetros dos sinais, conforme abordado a seguir.

3.3.3 Extração de Parâmetros

Os parâmetros escolhidos para estudar o nível de correlação com a rugosidade do processo foram o perfil de energia e os coeficientes mel-cepstrais.

Foi desenvolvido um software, utilizando linguagem C, para a extração dos parâmetros dos sons gravados, conforme exemplificado na Figura 3.8.

```
                pdMelCeps[quadro] = (double*)malloc(sizeof(double)*NUMBER_COEFICIENTES_MEL);
                if(!pdMelCeps[quadro])
                {
                    printf("\nError allocating memory");
                    return  -1;
                }
#else
                pdPerfilArray[quadro] = (double*)malloc(sizeof(double)*NUMBER_ENERGY_CONTOUR);
                if(!pdPerfilArray[quadro])
                {
                    printf("\nError allocating memory");
                    return  -1;
                }
#endif
                //Pega um quadro de 20ms
                for(iCounter=inicio;iCounter<fim;iCounter++)
                {
                    pdWindowedPre[iCounter-inicio]=pdPreEnfase[iCounter];
//                  pdWindowedPre[iCounter-inicio]=pdAudioData[iCounter];
                }

                //Multiplica pela janela de Hamming
                for(iCounter=0;iCounter<iHammingSize;iCounter++)
                    pdWindowedPre[iCounter] *= pdHamming[iCounter];

                //Calcula a FFT
                //fft_DATA contém o quadrado do módulo da FFT de cada janela de 20ms com tamanho fft_SIZE/2
                fft(pdWindowedPre, pdEnergy, M, iPointsFFT, iHammingSize);
```

Figura 3.8 - Trecho do código de extração de parâmetros.

Mais detalhes do software podem ser vistas no Anexo C. Este software foi utilizado para extrair os parâmetros de cada sinal e gravar em arquivo, gerando um banco de dados para cada parâmetro.

Para o perfil de energia, o número de níveis escolhido para este trabalho foi $\beta = 9$ o que representa uma distribuição da energia de 10% a 90%, conforme item 2.2.3. Para diminuir o número de pontos para análise e ainda suavizar a variação dos valores obtidos para o perfil, foi utilizado um filtro de média movente com uma janela de 30 amostras e passo entre janelas de 5 amostras. Utilizando o arquivo 'vc_186_f_015_ap_015-7.perfil' como exemplo, obteve-se um total de 194 quadros, onde cada quadro é formado por 9 níveis de perfil de energia. Dez quadros são apresentados na Figura 3.9.

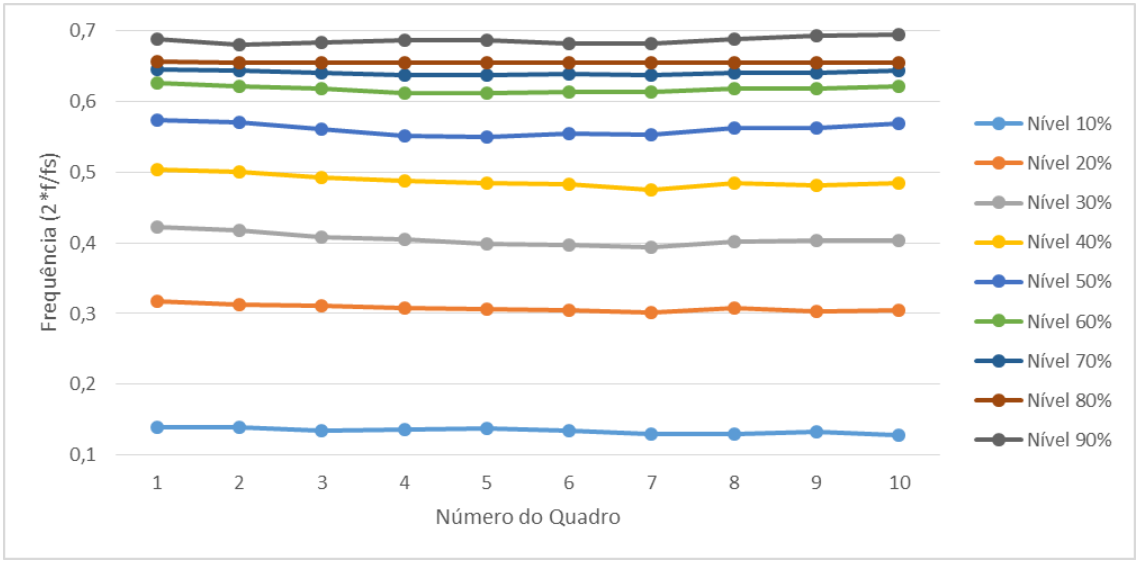


Figura 3.9 – Sequência com 10 quadros do perfil de energia.

Para os coeficientes mel-cepstrais, conforme mostrado no item 2.2.2, foram extraídos 12 coeficientes utilizando-se 32 filtros na escala Mel, já que a frequência de amostragem utilizada foi de 44,1kHz. Para o arquivo 'vc_186_f_015_ap_015-7.mel', obteve-se um total de 998 quadros, onde cada quadro é formado por 12 coeficientes mel-cepstrais. Dez quadros são apresentados na Figura 3.10.

De posse dos parâmetros dos sinais, o próximo passo é separar os conjuntos de treinamento e teste, conforme detalhado a seguir.

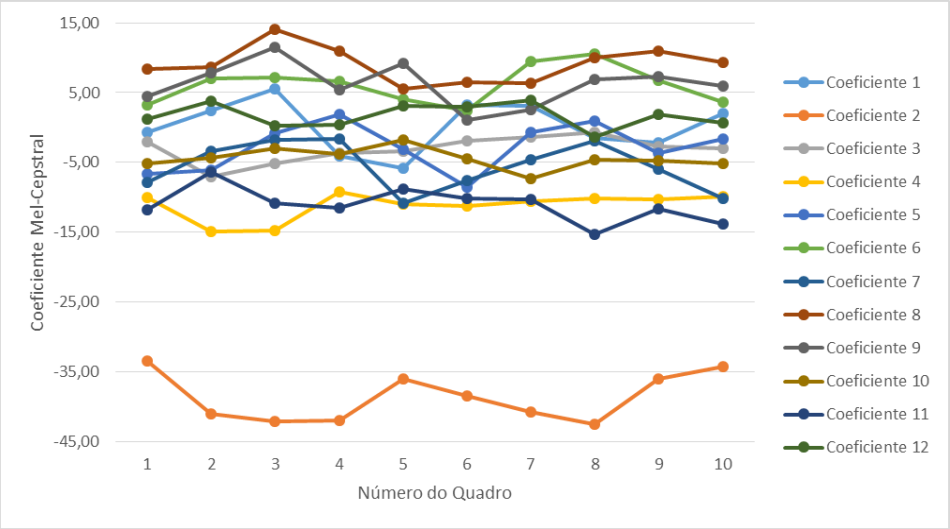


Figura 3.10 – Sequência com 10 quadros dos coeficientes mel-cepstrais.

3.4 Separação dos Grupos de Treinamento e Teste

Para reconhecer os padrões dos sinais acústicos e correlacioná-los com os valores obtidos de rugosidade, é necessário avaliar-se em quantas categorias distintas os valores medidos de rugosidade podem ser separados. A Figura 3.11 mostra a rugosidade média R_a medida em cada passo do processo. Cada valor de R_a mostrado na figura é uma média de 12 medidas efetuadas conforme especificado no item 3.3. A tabela com os dados que geraram a figura pode ser encontrada no Anexo A.

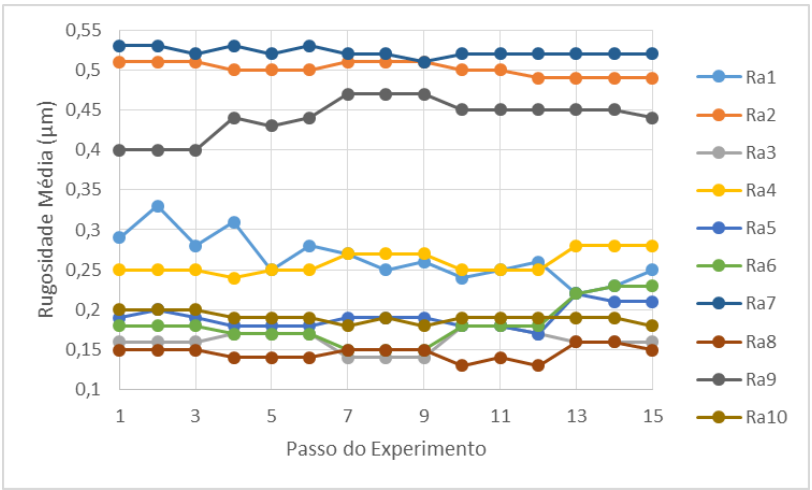


Figura 3.11 - Rugosidades médias resultantes do experimento.

Utilizando-se a técnica estatística de Análise Hierárquica de Cluster – Método de Ward (JOHSON E WICHERM, 2002), observou-se que algumas configurações experimentais não apresentam valores médios de rugosidade R_a suficientemente diferentes para suportar a hipótese de que seja possível separar os 10 conjuntos de parâmetros de usinagem. Desse modo, algumas configurações experimentais diferentes podem conduzir aos mesmos valores médios de rugosidade. A Figura 3.12 mostra como estes experimentos se agrupam assim como, em alguns casos, eles são perfeitamente separáveis formando 5 grupos distintos.

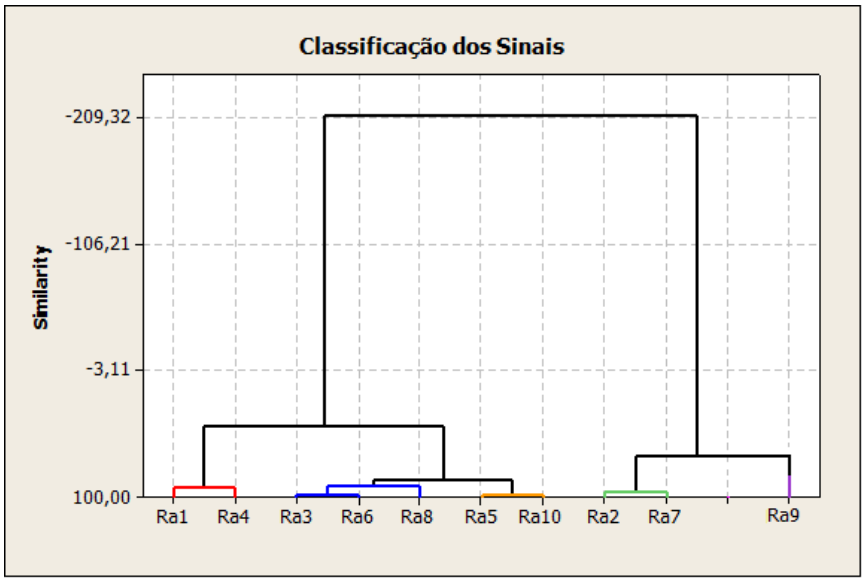


Figura 3.12 - Análise de *cluster* por experimento.

Nesta análise, avaliou-se a semelhança (*clusters*) entre os experimentos no que tange o comportamento de R_a. Alguns experimentos apresentam condições muito similares como, por exemplo, os experimentos 2 e 7 (*membership* 2) ou o grupo dos experimentos 3, 6 e 8 (*membership* 3). A Tabela 3.4 mostra a classificação dos experimentos em cinco clusters, de acordo com suas similaridades.

Tabela 3.4 - Separação dos grupos de rugosidade.

Experimento	Ra1	Ra2	Ra3	Ra4	Ra5	Ra6	Ra7	Ra8	Ra9	Ra10
Membership allocation	1	2	3	1	4	3	2	3	5	4

No caso dos experimentos 2 e 7, por exemplo, o *cluster* pode indicar uma dificuldade em se separar o comportamento da rugosidade com (V_c = 220, f = 0,23, a_p = 0,15) da condição (V_c = 240, f = 0,20, a_p = 0,20). Nota-se que os valores de R_a são realmente muito parecidos, o que dificultaria a separação das duas condições. O mesmo pode ser observado nos outros grupos formados.

Partindo-se da separação de *clusters* obtidas, definiram-se cinco grupos de rugosidade distintos. Portanto, para cada grupo, separaram-se dois conjuntos de treinamento e teste de acordo com a distribuição da Tabela 3.5. O segundo grupo de treinamento e teste foi utilizado como experimento de confirmação a fim de garantir a veracidade estatística dos dados.

Tabela 3.5- Conjuntos de treinamento e teste do classificador.

Grupo de Treinamento 1	**Grupo de Teste 1**	**Grupo de Treinamento 2**	**Grupo de Teste 2**
1	2	1	3
3	4	2	4
5	6	5	7
7	8	6	8
9	10	9	11
11	12	10	12
13	14	13	15
15		14	

Cada número da tabela representa o índice do arquivo com o parâmetro do sinal para cada configuração de usinagem ($V_{c,}, f, a_p$). Por fim, obteve-se um conjunto de oito arquivos para treinamento e sete arquivos para teste, para cada configuração. Os *clusters* 1, 2, e 4 são formados por dois conjuntos de configurações de usinagem o que totaliza dezesseis arquivos de treinamento e quatorze para teste. No caso do *cluster* 3, por possuir três conjuntos de configuração de usinagem, o número de arquivos de treinamento total é vinte e quatro e de teste vinte e um. Para o *cluster* 5, o número de arquivos de treinamento é oito e de teste, sete.

Os grupos formados serão utilizados na fase de treinamento e teste que serão detalhadas nos itens 3.5 e 3.6.

3.5 Treinamento de Grupos de Rugosidade

Os parâmetros extraídos dos sinais foram utilizados para treinar um GMM para cada grupo de rugosidade separado no item 3.4.

Para esta fase de treinamento, foi desenvolvido um software em linguagem C para implementar os algoritmos destacados no item 2.4. Este software utiliza os parâmetros de entrada para estimar um modelo com *N* gaussianas, com média e desvio padrão, que seja capaz de representar a sequência que o gerou. Mais detalhes do software podem ser vistos no Anexo D.

```
while(dist > 1e-5)
{
    // Iniciando contagem de tempo de processamento
    time(&t_inic);

    // Updating mixture coefficients
    update_p(x,dim,nFrames,l,nGaussians,p);

    // Updating gaussian means
    update_m(x,dim,nFrames,l,nGaussians,&m);

    // Updating gaussian variances
    update_s(x,dim,nFrames,l,nGaussians,m,&s);

    // Copying updated values for the model
    for (i=0;i<nGaussians;i++)
    {
        l[i].p = p[i];
        for (j=0;j<dim;j++)
        {
        l[i].m[j] = m[i][j];
            l[i].s[j] = s[i][j];
        }
    }

    // Verifying convergence
    p_new = testgmm(x,dim,nFrames,l,nGaussians);
    dist = -(p_new-p_old)/p_new;
```

Figura 3.13 - Fragmento do código de treinamento de GMM.

Foram utilizados modelos com 4 e 8 gaussianas para uma comparação de desempenho. Como o software aceita apenas a entrada de uma sequência de treinamento por vez, foi utilizada o fluxograma da Figura 3.14 para geração dos modelos GMM.

De acordo com o fluxograma, cada grupo de rugosidade possui uma sequência de treinamento, que é formada pelos parâmetros extraídos dos sinais de treinamento, para cada grupo de rugosidade. Esta sequência é apresentada ao software de treinamento, gerando um GMM para seu modelamento estatístico. Este fluxo repete-se para cada grupo de rugosidade.

Este processo foi utilizado para geração de duas bases de modelos GMM, uma para MFCC e outra para perfil de energia, que foram utilizados posteriormente para o teste de grupos de rugosidade. Mais detalhes dos resultados obtidos na fase de treinamento podem ser vistos no Anexo A.

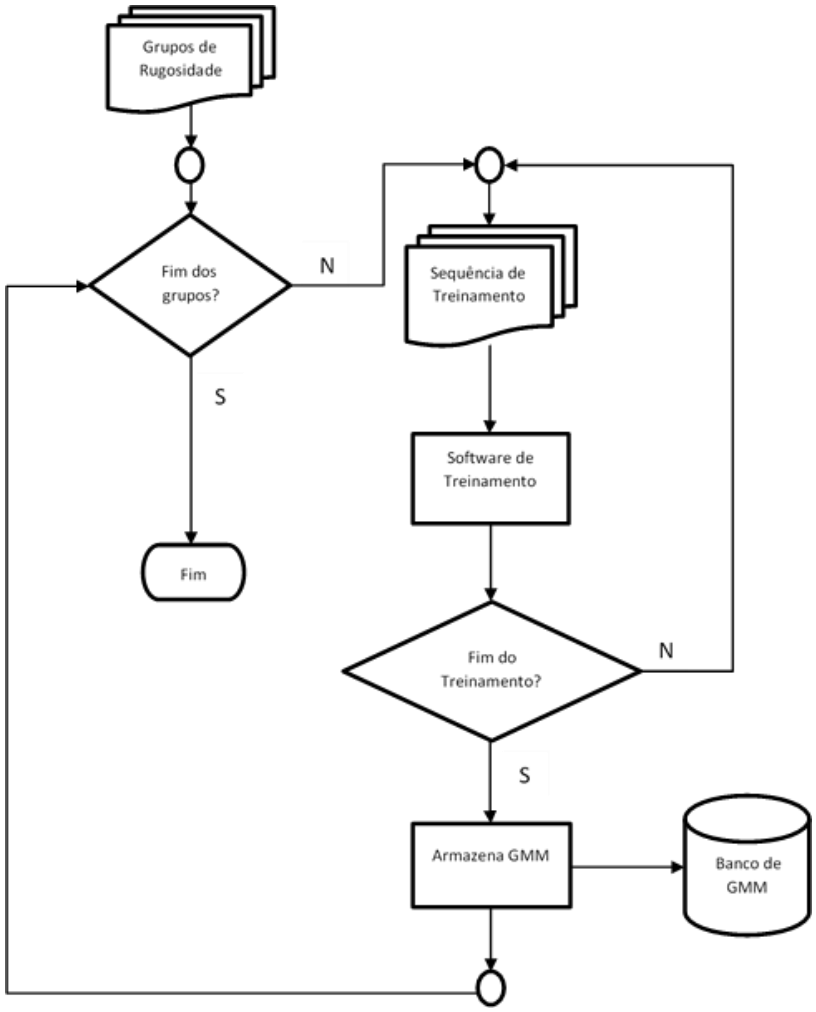


Figura 3.14 – Fluxograma do processo de treinamento dos modelos GMM.

3.6 Teste de Grupo de Rugosidade

Para a classificação, também foi desenvolvido um software em linguagem C onde, a sequência de observação dos parâmetros do sinal de teste é comparada a um GMM, retornando a probabilidade de este modelo representá-lo, representado por P(O|GMM). Mais detalhes do software podem ser vistos no Anexo E.

Este processo deve ser repetido para cada GMM gerado no processo de treinamento (item 3.5). O grupo de rugosidade ao qual o sinal de teste pertence é decidido baseado na maior probabilidade de saída P(O|GMM) obtida entre todos os modelos GMM, calculado conforme apresentado no item 2.5.

```
loadGMM(nGaussians,dim,&l,GMMFilename);

// Showing GMM in the screen
//showgmm(l,nGaussians,dim);

// Reading utterance to be verified
loadUtterance(utteranceFilename,dim,&nFrames,&x);

printf("Number of frames: %d\n",nFrames);
//for (t=0;t<nFrames;t++)
//{
 //   for(i=0;i<dim;i++)
  //      printf("%f\t",x[t][i]);
  //  puts(" ");
//}

p = aposteriori(x,dim,nFrames,l,nGaussians);

printf("P(O|GMM)=%f\n",p);

// Deallocating memory
// GMM
for (i=0;i<nGaussians;i++)
{
  free(l[i].m);
  free(l[i].s);
}
free(l);
```

Figura 3.15 - Fragmento do código de teste do grupo de rugosidade.

Se a probabilidade P(O|GMM) que apresentar o maior valor corresponder ao grupo GMM ao qual pertence o sinal de teste, o sistema terá identificado o grupo de rugosidade corretamente. Desta forma, o desempenho do sistema pode calculado através do quociente entre o número de acertos e número de testes executados, para cada grupo de teste.

$$Desempenho\ \% = \frac{n^{O}\,de\ sinais\ identificados\ corretamente}{n^{O}\ de\ sinais\ utilizados\ para\ teste} \times 100 \qquad (3.1)$$

O processo de teste dos grupos de rugosidade pode ser representado conforme fluxograma da Figura 3.16.

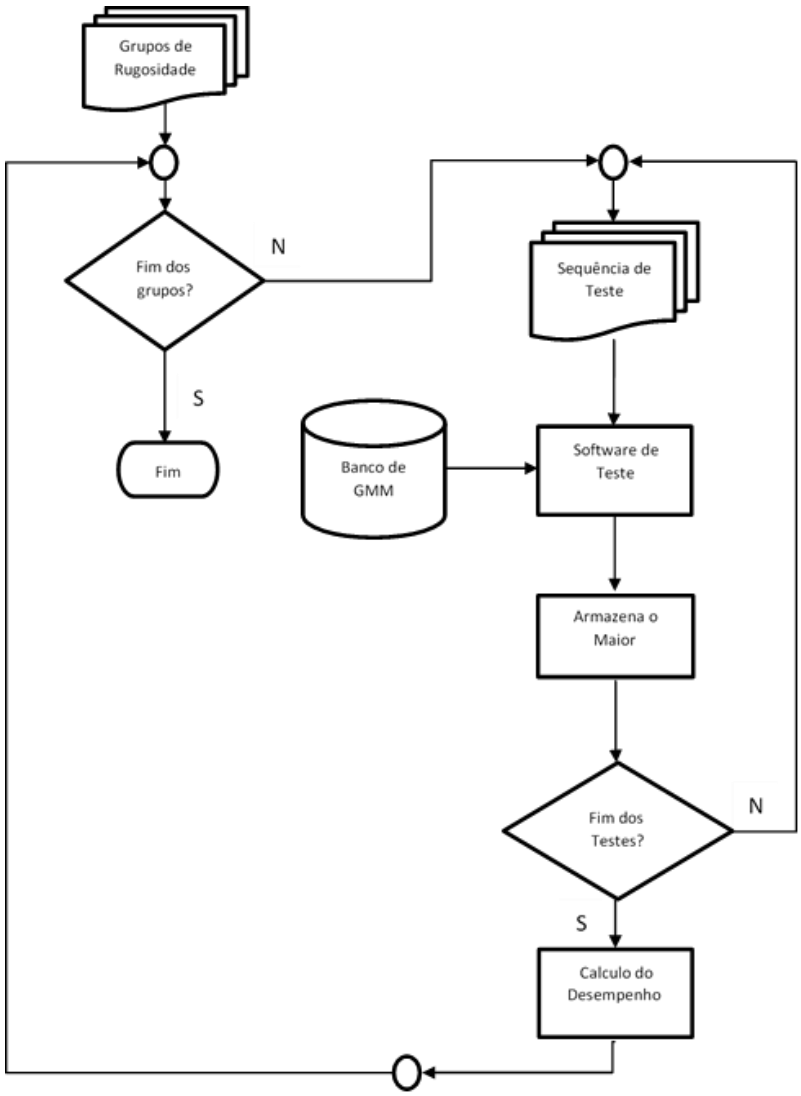


Figura 3.16 - Fluxograma do processo de teste para os grupos de rugosidade.

As tabelas com os resultados da fase de testes, para perfil de energia e para MFCC, podem ser encontradas no Anexo A. Os testes foram executados para os modelos treinados com 4 gaussianas e 8 gaussianas.

4. Resultados e discussão

Através do planejamento experimental discutido anteriormente, obteve-se ao fim da fase de treinamento um banco formado por 5 modelos GMM baseados no MFCC e 5 modelos baseados no perfil de energia. Cada modelo representa um grupo de rugosidade, conforme identificados no item 3.4.

A Tabela 4.1 apresenta dois modelos GMM com 4 gaussianas, um modelo para R_{a1} e outro para $R_{a4,}$ treinados por uma sequência de observação com perfil de energia como parâmetro. De acordo com o número de níveis escolhidos no item 3.3.3, obteve-se 9 médias e 9 desvios-padrão para representá-la. Cada gaussiana treinada tem um peso identificado por *p*, onde a somatória de todos os pesos deve ser igual a 1.

Tabela 4.1- Modelo GMM treinado com 4 gaussianas para perfil de energia.

Gaussian 1: p = 0,222568	Gaussian 2: p = 0,257042	Gaussian 3: p = 0,261373	Gaussian 4: p = 0,259017	Gaussian 1: p = 0,196547	Gaussian 2: p = 0,273797	Gaussian 3: p = 0,267653	Gaussian 4: p = 0,262003
m =	m =	m =	m =	m =	m =	m =	m =
0,157897	0,158983	0,160170	0,161037	0,182741	0,200190	0,208264	0,210020
0,290509	0,294581	0,297044	0,298336	0,304171	0,326469	0,335935	0,337907
0,394207	0,399700	0,402424	0,403705	0,374878	0,401391	0,412467	0,414741
0,485044	0,491228	0,493938	0,495142	0,433918	0,46213	0,473689	0,476026
0,558777	0,564653	0,566949	0,567931	0,474634	0,501494	0,512386	0,514571
0,604395	0,609068	0,610688	0,611362	0,511343	0,534771	0,544245	0,546135
0,631359	0,634462	0,635442	0,635850	0,564581	0,582012	0,588783	0,590099
0,654475	0,656202	0,656773	0,657061	0,616826	0,62671	0,630388	0,631088
0,741567	0,743242	0,744005	0,744557	0,697561	0,70477	0,707905	0,708619
s =	s =	s =	s =	s =	s =	s =	s =
0,004068	0,004138	0,004168	0,004195	0,007985	0,007998	0,008006	0,008017
0,006498	0,006193	0,00602	0,005938	0,012127	0,010150	0,009464	0,009355
0,007530	0,006742	0,006435	0,006303	0,016850	0,013608	0,012510	0,012335
0,008409	0,007158	0,006732	0,006557	0,019393	0,014981	0,013474	0,013225
0,007392	0,005940	0,005484	0,005300	0,017811	0,013468	0,011989	0,011743
0,005071	0,003855	0,003502	0,003360	0,013615	0,010296	0,009153	0,008962
0,002730	0,001977	0,001777	0,001696	0,008250	0,005824	0,004985	0,004842
0,001170	0,001000	0,001000	0,001000	0,002928	0,001917	0,001575	0,001518
0,003555	0,003467	0,003451	0,003451	0,002321	0,002287	0,002292	0,002300

Já na Tabela 4.2, os modelos GMM com 4 gaussianas foram treinados por uma sequência de observação com parâmetro MFCC. De acordo com o número de coeficientes escolhidos no item 3.3.3, obteve-se 12 médias e 12 desvios-padrão.

Tabela 4.2 - Modelo GMM treinado com 4 gaussianas para MFCC.

Gaussian 1: p = 0,196513	Gaussian 2: p = 0,319758	Gaussian 3: p = 0.,42047	Gaussian 4: p = 0,141681	Gaussian 1: p = 0,365029	Gaussian 2: p = 0,052048	Gaussian 3: p = 0,086734	Gaussian 4: p = 0,496189
m =	m =	m =	m =	m =	m =	m =	m =
4,708660	2,559635	16,772171	16,317800	11,245571	-25,112029	32,444528	25,621582
-36,570090	-37,561852	-42,294478	-39,534124	-45,608465	-14,683255	-46,527253	-45,175695
-1,662207	-0,884334	-0,419470	-2,366309	2,187158	-3,140880	-7,344191	-5,192507
-14,057479	-9,924318	-8,202229	-1,058866	-7,608743	-11,220783	-5,041556	1,510427
-6,626214	-1,532658	-1,343543	3,663393	1,111801	0,538717	-4,377055	-0,246815
1,791225	5,029689	4,738661	2,766217	6,097667	11,342352	6,677413	6,246771
-8,047162	-7,015262	-4,918850	-5,335950	-8,069549	-0,168412	-1,006964	-7,197230
8,477226	6,642771	6,477452	6,988379	4,693062	10,632627	8,561898	3,357175
6,564075	3,549046	3,882075	1,280410	2,044115	5,104249	4,622452	3,787283
-2,135235	-4,205813	-4,400439	-4,204862	-1,299712	-7,058413	-6,565772	-0,945557
-10,736993	-9,912029	-7,764701	-1,585619	-6,211816	-7,423108	-9,041466	-3,750326
-0,103372	2,204314	2,068337	6,225166	3,840753	5,44274	-0,72080	1,35663
s =	s =	s =	s =	s =	s =	s =	s =
28,771796	30,396104	23,310859	33,144493	38,079095	27,342723	24,789814	22,093228
15,346823	17,090161	26,720324	15,640801	17,805978	29,141068	9,095208	14,473660
7,877418	11,774272	8,981816	10,866782	9,219463	10,393923	6,796680	7,455327
7,765884	9,272485	10,546270	16,680496	13,271186	5,984327	6,914750	9,226887
5,876643	8,410774	8,331390	7,805347	10,318700	5,146251	6,244597	7,828917
6,439358	7,940300	7,412359	13,249440	7,042203	6,682759	6,447666	7,177667
8,097958	12,169296	10,995683	6,415826	11,347243	5,585496	6,740599	7,233339
9,360952	7,969687	9,849259	9,710343	11,221713	5,808581	7,719720	7,184835
7,517936	7,555373	6,499283	7,182764	7,709262	5,222407	6,627153	7,662990
7,678815	10,135500	8,071458	10,045075	10,416980	5,025901	7,100851	7,377227
7,390923	9,632738	8,032347	8,692604	10,963885	4,491162	4,897287	8,294498
7,529280	8,793852	6,060492	5,298816	5,974196	5,014568	4,291485	5,012848

Ao utilizar os modelos resultantes do treinamento para o teste dos grupos de rugosidade, obtiveram-se os melhores desempenhos com os modelos treinados pelo parâmetro MFCC, conforme pode ser visto na Figura 4.1. O cálculo do desempenho foi baseado na equação (3.1).

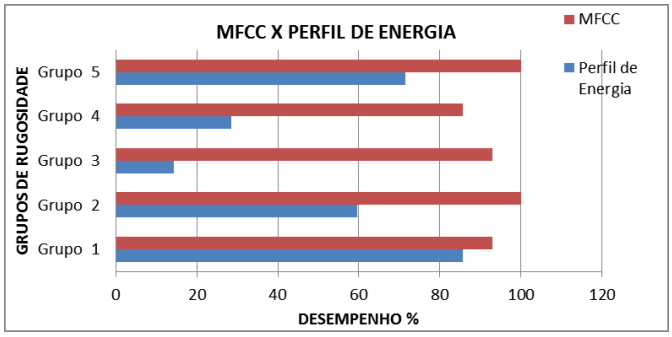
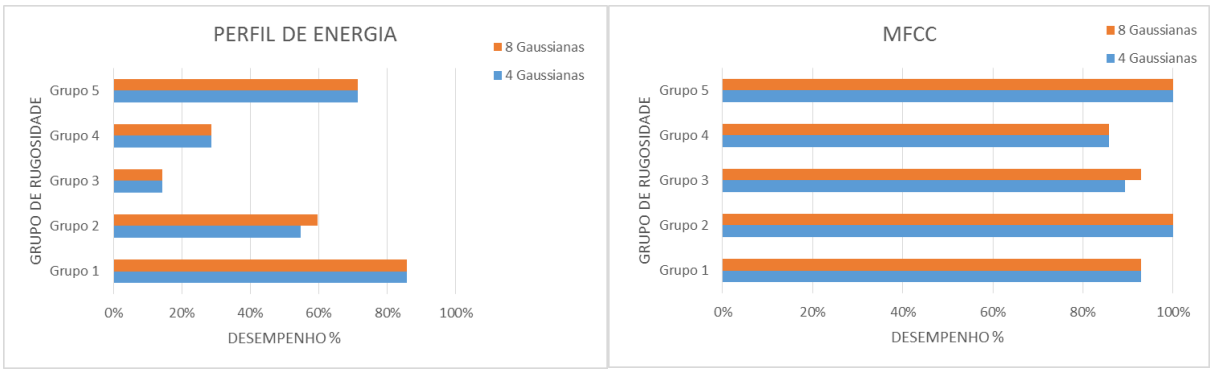


Figura 4.1 - Desempenho do classificador para GMM com 8 gaussianas.

Ao analisar a Tabela 4.1 para perfil de energia, nota-se as quatro gaussianas resultantes para o grupo de rugosidade R_{a1} apresentaram médias muito próximas das gaussianas resultantes para R_{a4}. Esta proximidade aumenta a probabilidade do classificador confundir os grupos. Outro fator que impacta no desempenho do classificador com perfil de energia é o pequeno valor do desvio-padrão obtido. Isto resulta em erros de classificação se houver pequenas variações de valor durante a sequencia de teste. Este problema não é observado na Tabela 4.2 para MFCC onde as gaussianas resultantes apresentam médias distantes um das outras assim como médias diferentes de um grupo de rugosidade para outro.

Desta forma, pode-se verificar nos resultados que o MFCC apresenta uma correlação maior com a rugosidade em relação ao perfil de energia, onde o sistema consegue identificar o nível de rugosidade com maior precisão, apesentando um desempenho médio na ordem de 94,29% enquanto o perfil de energia apresentou um desempenho médio de apenas 51,90%.

Já o aumento do número de gaussianas não apresentou melhora significativa no reconhecimento do padrão de rugosidade. Uma comparação entre os sistemas treinados com 4 gaussianas e 8 gaussianas é apresentado na Figura 4.2. O aumento do número de gaussianas apresentou uma melhora no desempenho para apenas um grupo de rugosidade, de 4,76% para perfil de energia e 3,57% para MFCC. Para os demais grupos de rugosidade não houve melhora de desempenho.



(a) Perfil de energia (b) MFCC

Figura 4.2 - Desempenho variando o número de gaussianas.

De forma geral, os resultados obtidos por grupo de rugosidade, por número de gaussianas e por parâmetro do sinal, podem ser resumidos conforme Figura 4.3, que apresenta

um comparação estatística entre os resultados através de uma análise de variância (ANOVA) e da Figura 4.4, que apresenta o gráfico fatorial.

ANOVA: Tx. Acerto versus Grupo; No. Gaussianas; Método

Factor	Type	Levels	Values
Grupo	fixed	5	1; 2; 3; 4; 5
No. Gaussianas	fixed	2	4Gauss; 8Gauss
Método	fixed	2	MFCC; Perfil de Energia

Analysis of Variance for Tx. Acerto

Source	DF	SS	MS	F	P
Grupo	4	0.44869	0.11217	5.12	0.011
No. Gaussianas	1	0.00035	0.00035	0.02	0.902
Método	1	0.90317	0.90317	41.23	0.000
Error	13	0.28478	0.02191		
Total	19	1.63698			

S = 0.148008 R-Sq = 82.60% R-Sq(adj) = 74.57%

Figura 4.3 - Anova para taxa de acerto.

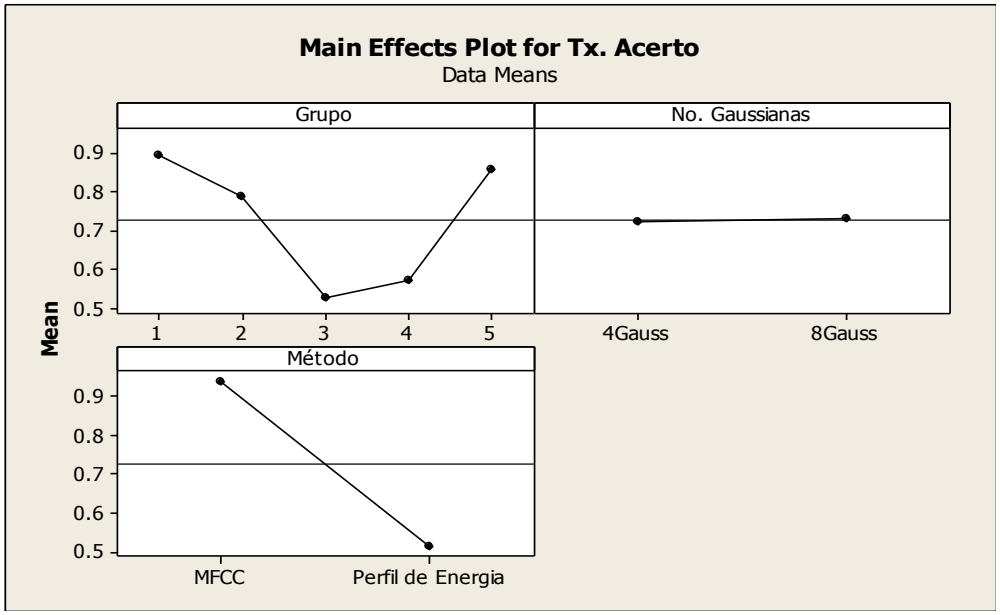


Figura 4.4 - Gráficos fatoriais para taxa de acerto.

A Figura 4.5 e a Figura 4.6 apresentam uma análise estatística para a probabilidade de saída P(O|GMM) para o modelo com 4 gaussianas utilizando perfil de energia e MFCC respectivamente.

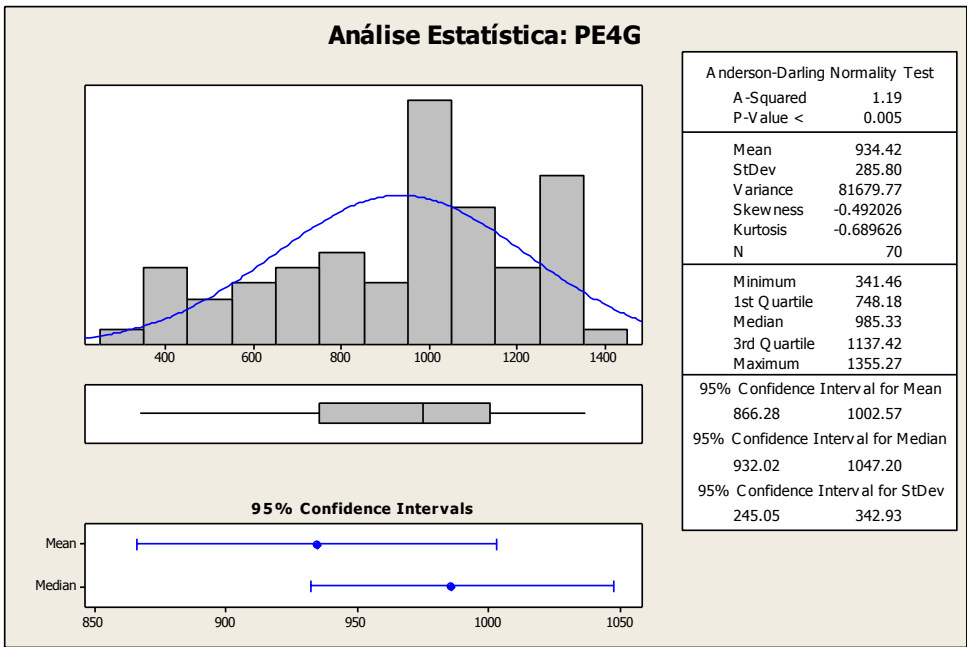


Figura 4.5 – Análise Estatística de P(O|GMM) - Perfil de Energia e 4 gaussianas.

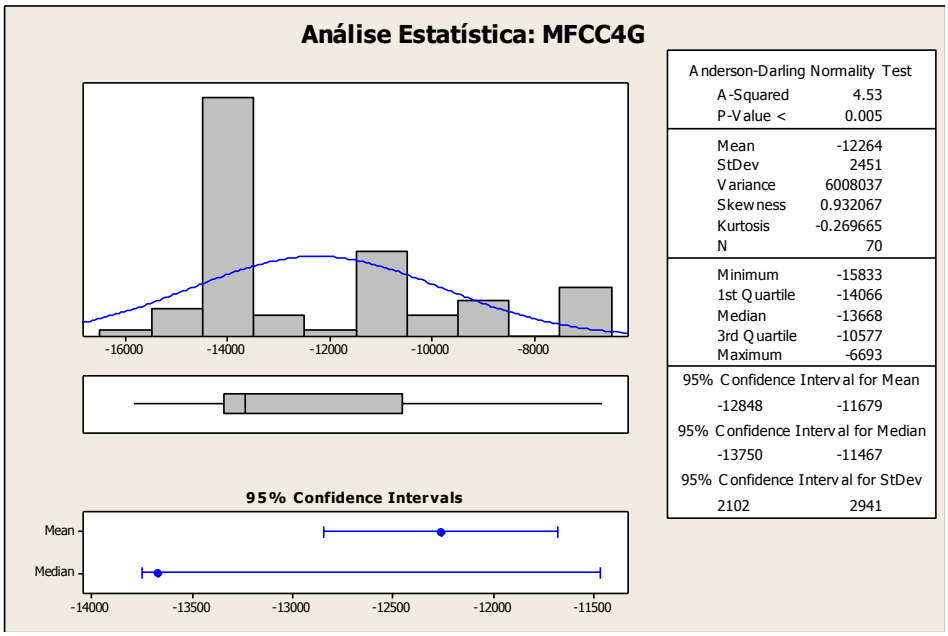


Figura 4.6 – Análise Estatística de P(O|GMM) - MFCC e 4 gaussianas.

Já a Figura 4.7 e a Figura 4.8 apresentam uma análise estatística para a probabilidade de saída P(O|GMM) para o modelo com 8 gaussianas utilizando perfil de energia e MFCC respectivamente.

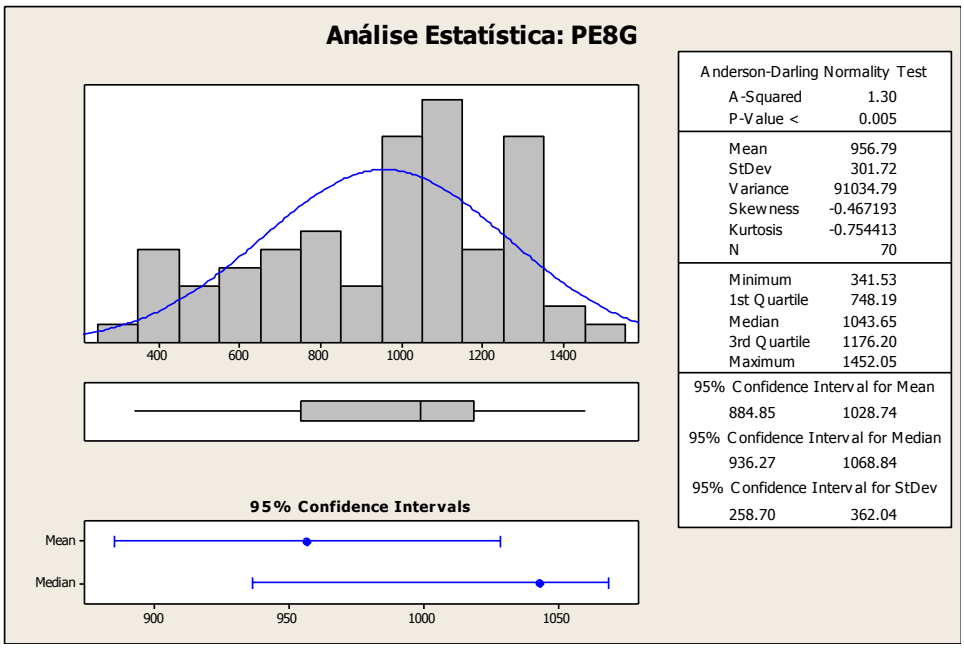


Figura 4.7 – Análise Estatística de P(O|GMM) - Perfil de Energia e 8 gaussianas.

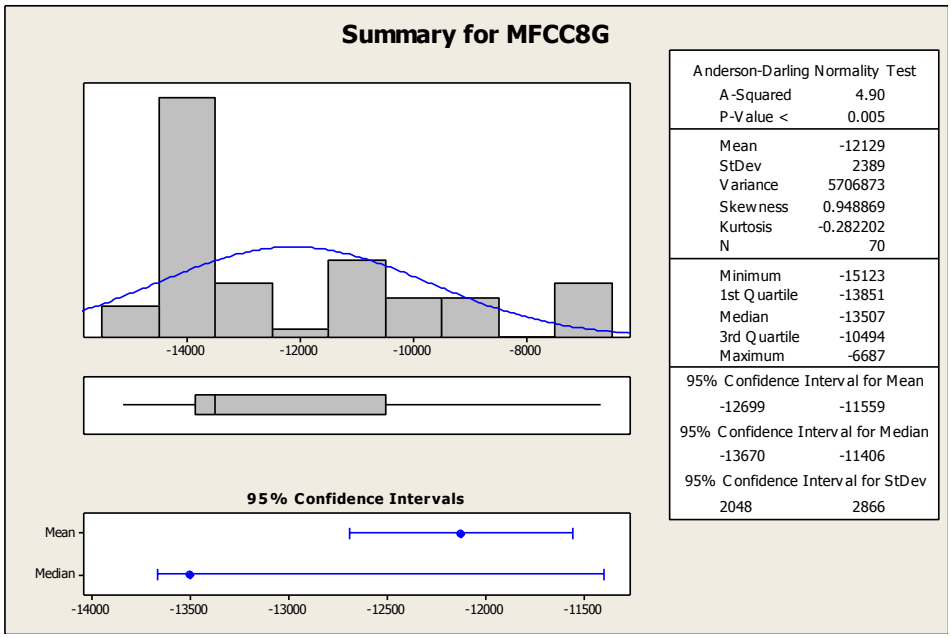


Figura 4.8 – Análise Estatística de P(O|GMM) - MFCC de 8 gaussianas.

A Figura 4.9 apresenta o nível de separação entre os diferentes grupos de rugosidade a partir do modelo com 4 gaussianas treinado com perfil de energia.

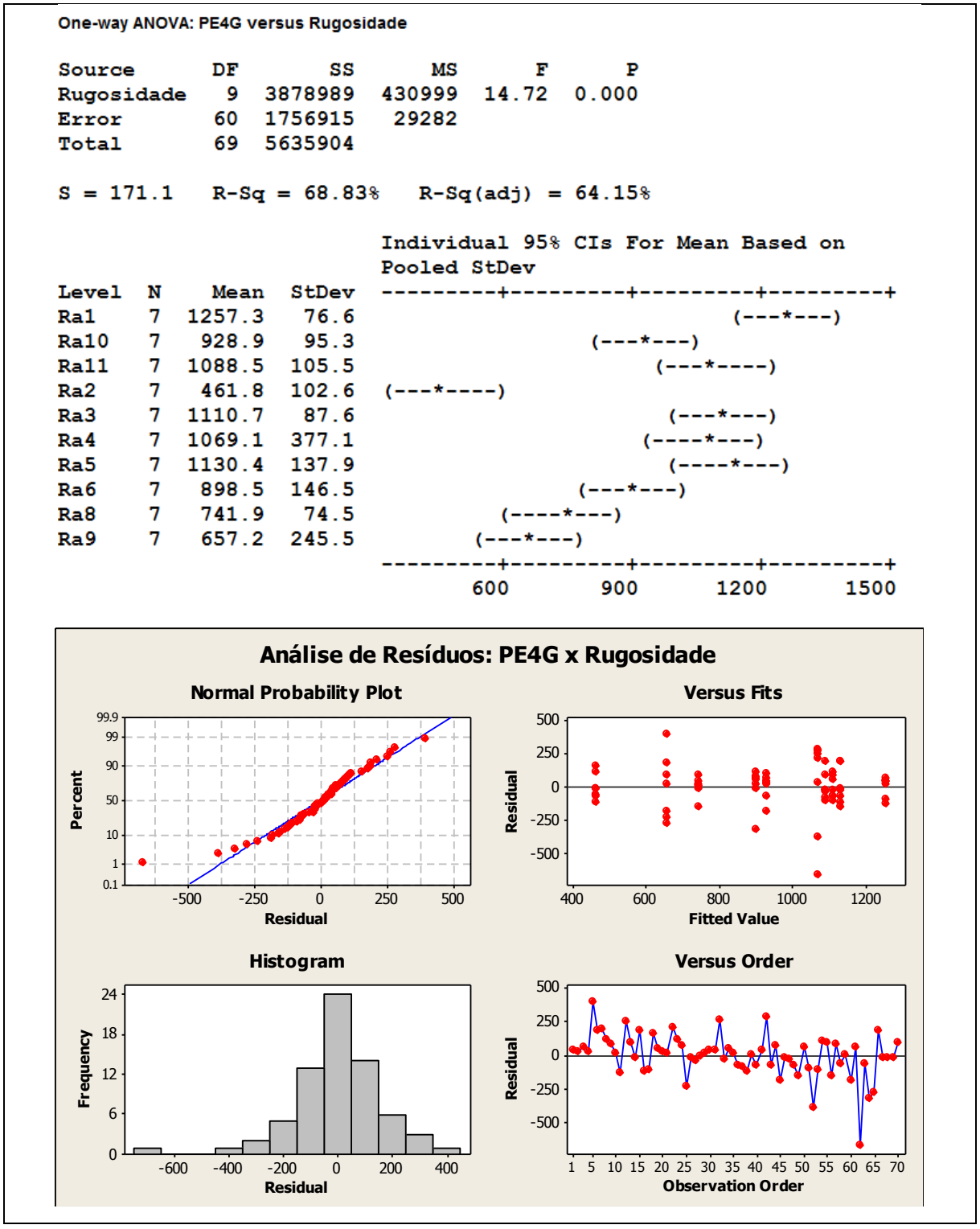


Figura 4.9 – Relação entre P(O|GMM) - PE4G e os níveis de rugosidade.

A Figura 4.10 apresenta o nível de separação entre os diferentes grupos de rugosidade a partir do modelo com 8 gaussianas treinado com perfil de energia.

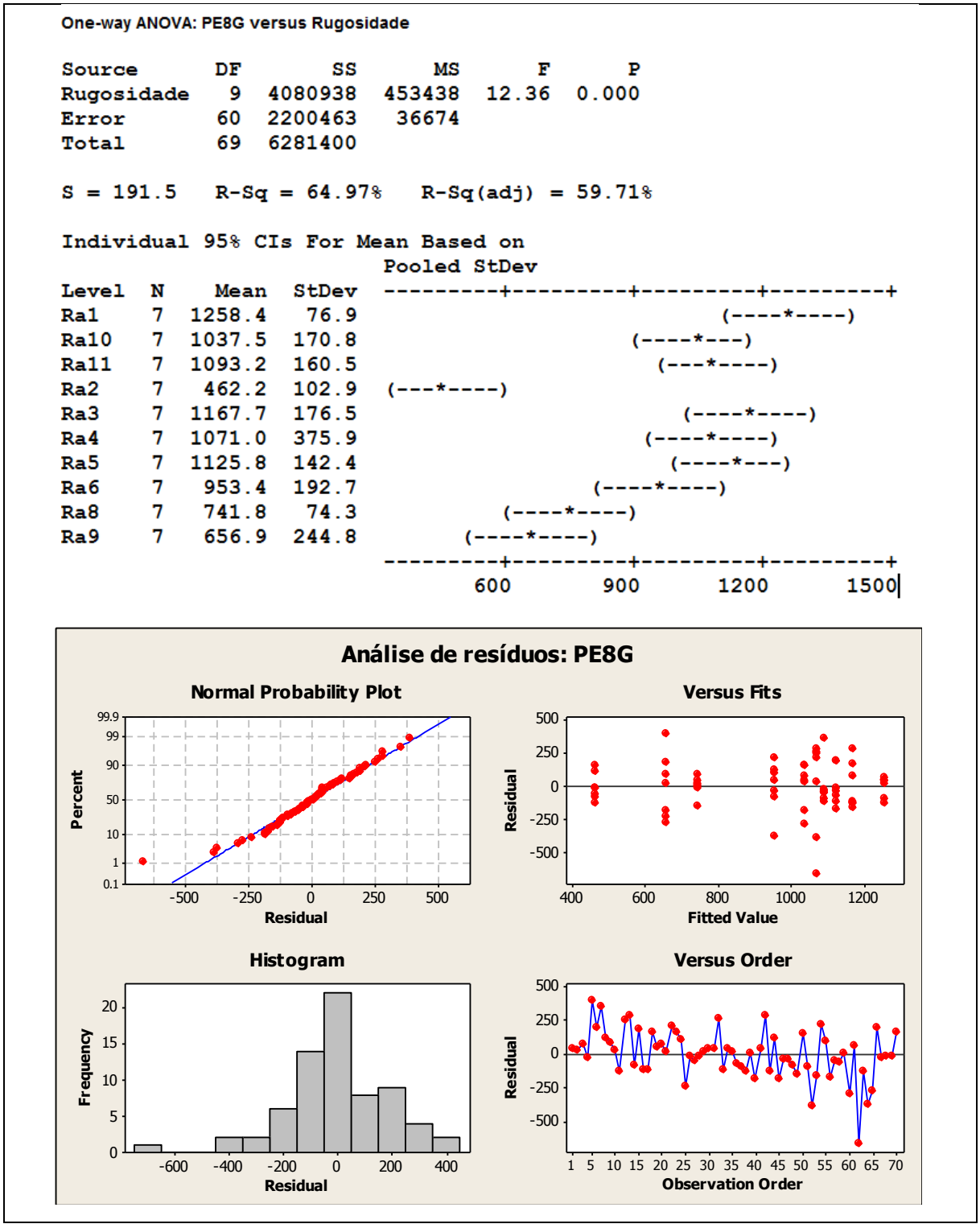


Figura 4.10 – Relação entre P(O|GMM) - PE8G e os níveis de rugosidade.

A Figura 4.11 apresenta o nível de separação entre os diferentes grupos de rugosidade a partir do modelo com 4 gaussianas treinado com MFCC.

One-way ANOVA: MFCC4G versus Rugosidade

Source	DF	SS	MS	F	P
Rugosidade	9	401652431	44628048	207.54	0.000
Error	60	12902125	215035		
Total	69	414554556			

S = 463.7 R-Sq = 96.89% R-Sq(adj) = 96.42%

Level	N	Mean	StDev	Individual 95% CIs For Mean Based on Pooled StDev
Ra1	7	-13872	338	(-*)
Ra10	7	-10652	395	(-*)
Ra11	7	-14233	399	(-*-)
Ra2	7	-7004	232	(-*-)
Ra3	7	-13868	207	(-*)
Ra4	7	-13998	287	(-*-)
Ra5	7	-13955	845	(-*-)
Ra6	7	-11071	296	(-*)
Ra8	7	-9685	710	(-*)
Ra9	7	-14298	487	(-*)

-14000 -12000 -10000 -8000

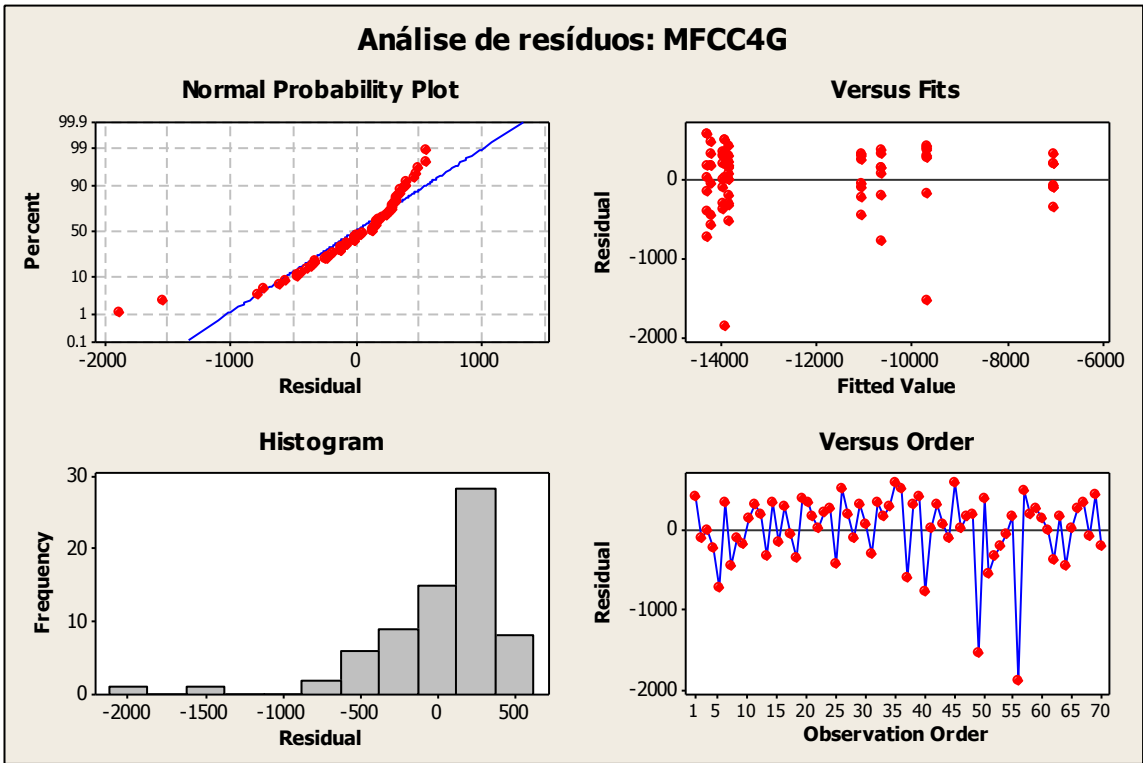


Figura 4.11– Relação entre P(O|GMM) - MFCC4G e os níveis de rugosidade.

A Figura 4.12 apresenta o nível de separação entre os diferentes grupos de rugosidade a partir do modelo com 8 gaussianas treinado com MFCC.

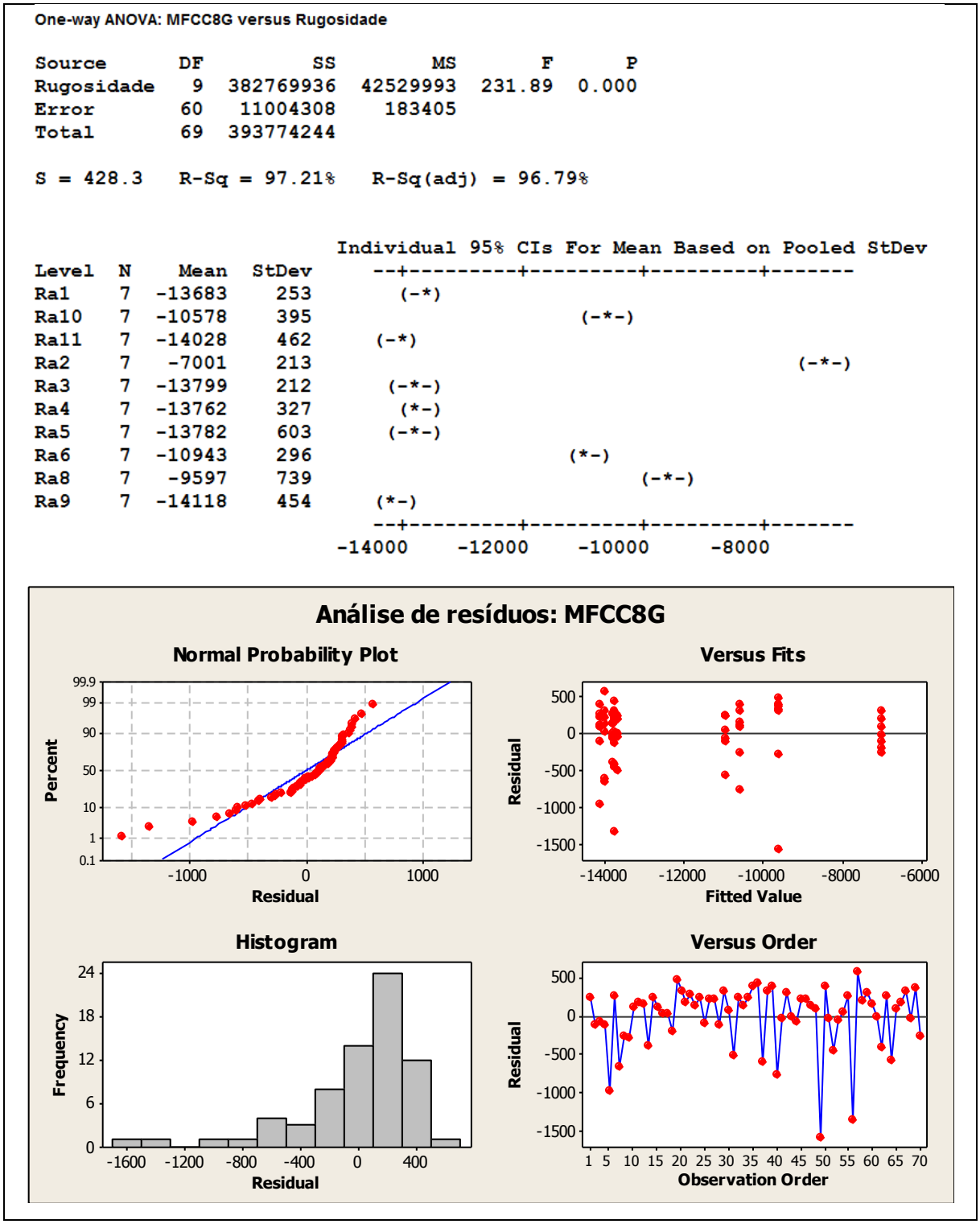


Figura 4.12 – Relação entre P(O|GMM) - MFCC8G e os níveis de rugosidade

Utilizando a Figura 4.13, que apresenta a curva de superfície de resposta para MFCC com 8 gaussianas correlacionados com os parâmetros de entrada, e a curva de superfície de resposta da rugosidade Ra pelos mesmos parâmetros de entrada (avanço *f* e profundidade de corte *ap*), pode-se observar a semelhança entra a superfície obtida para o MFCC e a superfície da rugosidade Ra. Esta semelhança mostra que, além de existir uma correlação entre o sinal acústico e a rugosidade, esta correlação também existe com os parâmetros de corte.

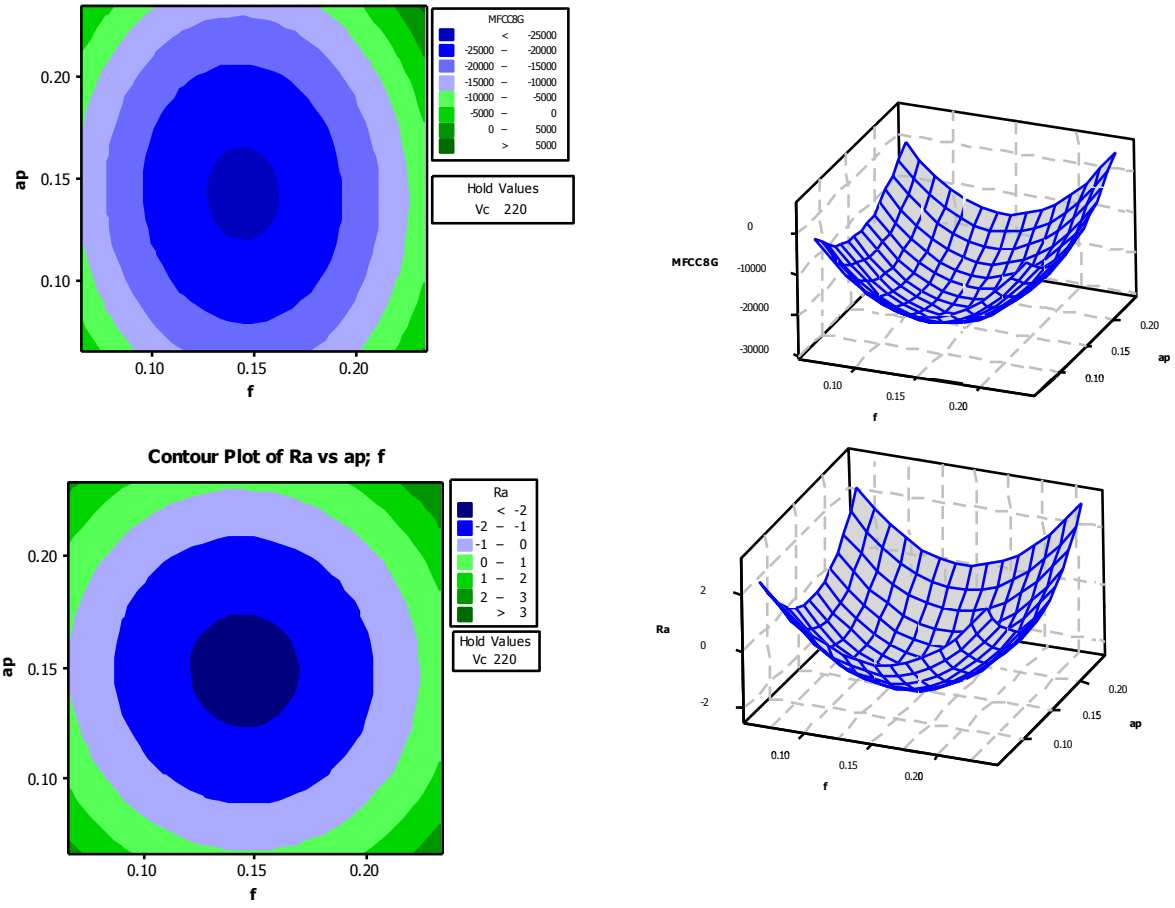


Figura 4.13 – Superfícies de Resposta para MFCC8G e Ra.

Esta correlação entre parâmetros de corte e o sinal acústico também pode ser observada na Figura 4.14, que apresenta a análise estatística entre a probabilidade do modelo (com 8 gaussianas treinado com MFCC) representar a sequência de teste, representado por P(O|GMM), e os parâmetros de corte (avanço *f* e profundidade de corte *ap*).

Response Surface Regression: MFCC8G* versus Vc; f; ap

The analysis was done using coded units.

Estimated Regression Coefficients for MFCC8G

Term	Coef	SE Coef	T	P
Constant	-25671.7	3376.66	-7.603	0.000
Vc	-29.6	80.93	-0.366	0.716
f	1221.5	80.93	15.092	0.000
ap	1068.5	80.93	13.202	0.000
Vc*Vc	4221.0	1147.15	3.680	0.001
f*f	5874.7	1147.15	5.121	0.000
ap*ap	3575.4	1147.15	3.117	0.003
Vc*f	-880.9	80.93	-10.884	0.000
Vc*ap	691.5	80.93	8.545	0.000
f*ap	332.1	80.93	4.103	0.000

S = 428.258 PRESS = 14978086
R-Sq = 97.21% R-Sq(pred) = 96.20% R-Sq(adj) = 96.79%

Figura 4.14 – Análise estatística do *D-optimal design* para P(O|GMM) - MFCC8G.

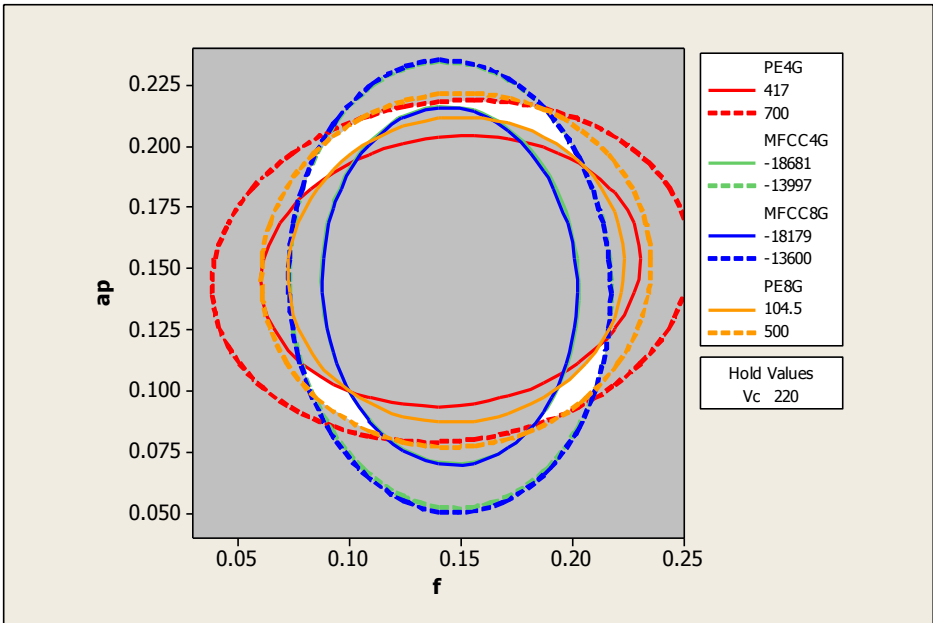


Figura 4.15 – Gráfico de sobreposição das Superfícies de Resposta.

Na Figura 4.15 é possível identificar uma superposição entre as superfícies de resposta para os modelos com 4 e 8 gaussianas treinados com perfil de energia e MFCC.

Já a Figura 4.16 e a Figura 4.17 apresentam a relação entre as rugosidades e a probabilidade P(O|GMM) obtidas para o modelo com 8 gaussianas treinado com MFCC.

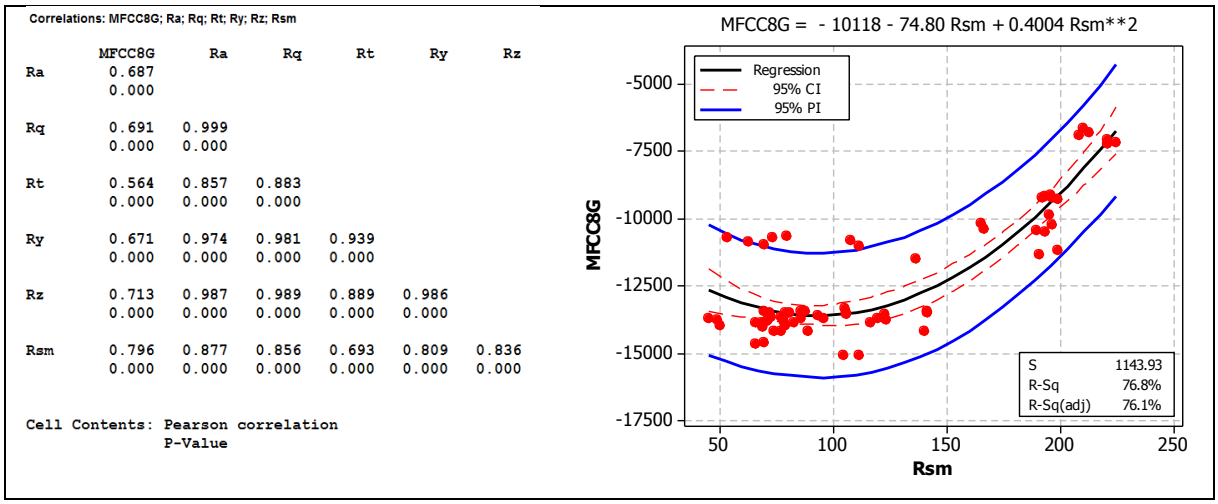


Figura 4.16– Relação entre as Rugosidades e P(O|GMM) - MFCC8G.

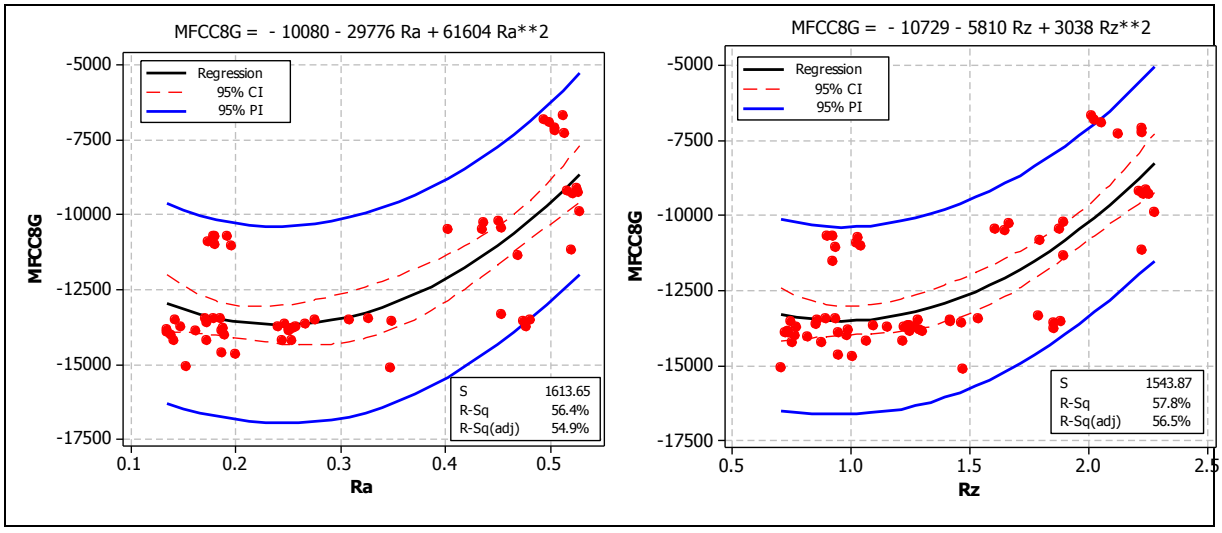


Figura 4.17 – Relação entre as Rugosidades Ra, Rz e P(O|GMM) - MFCC8G.

5. Conclusões

Neste trabalho, foi proposto um método de monitoramento do processo de torneamento utilizando sinal acústico e modelos de misturas de gaussianas. Para garantir a viabilidade do método, seria necessário obter uma comprovação estatística de que primeiramente, o sinal acústico poderia ser utilizado como fonte de informações sobre o fenômeno físico, no caso a rugosidade resultante do processo, e que o modelamento estatístico proposto fosse capaz de identificar estas informações. Baseando-se nestas informações e nos resultados obtidos, pode-se concluir que:

- Os sinais acústicos apresentaram características capazes de parametrizar a rugosidade resultante do processo em 5 níveis diferentes. Foi possível extrair dois parâmetros do sinal (perfil de energia e MFCC), cada um com características diferentes;
- Dentre os parâmetros estudados, o MFCC apresentou a maior correlação com os valores de rugosidade medidas, apresentando desempenho superior ao modelo utilizando perfil de energia;
- O modelo utilizando GMM apresentou desempenho médio na ordem de 94,27% considerando os 5 grupos de rugosidade e o MFCC como parâmetro;
- O aumento do número de gaussianas de 4 para 8 não melhorou o desempenho de forma significativa;
- A utilização do microfone como meio de captura de sinais para monitoramento mostrou-se eficiente comparando-se o custo do mesmo em relação aos sensores de emissão acústica (acelerômetros), considerando-se os resultados obtidos.

Com os resultados obtidos é possível concluir que o modelo proposto consegue identificar diferentes níveis de rugosidade do processo analisando o som emitido durante a usinagem. Portanto, o modelo pode ser utilizado como um método de monitoramento indireto.

Outro resultado importante é a consolidação do MFCC como parâmetro para o monitoramento de processos de torneamento duro, a partir do sinal acústico.

5.1 Sugestões para Trabalhos Futuros

- Monitorar outros processos de manufatura, tais como, soldagem, fresamento, retificação, furação, etc. poderiam ser analisados utilizando o método proposto;
- Testar o MFCC em diferentes processos de fabricação;
- Outras técnicas de modelamento estatístico poderiam ser avaliadas e comparadas como, por exemplo, as Redes Neurais, Modelos Ocultos de Markov, entre outras;
- Pode-se extrair outros parâmetros do sinal acústico;
- Outras combinações com diferentes materiais, equipamentos ou ferramentas podem ser utilizadas para novas avaliações;
- Pode-se avaliar o monitoramento do desgaste de ferramenta através do método proposto neste trabalho;
- Estudar a influência do ruído provocado por outros tornos no método proposto (utilização de filtros baseados em vários microfones);
- Estudar a influência de ruídos de outros processos no método proposto (utilização de filtros digitais);
- Estudar uma forma de aumentar o desvio-padrão para os modelos obtidos utilizando perfil de energia e reavaliar os resultados;
- Reavaliar a frequência de amostragem utilizada para gravação dos sinais de áudio;
- Considerar os quatro estágios do sinal (usando os trechos com a entrada e a saída da ferramenta na peça) utilizando HMM;
- Avaliar a possibilidade de se considerar também as covariâncias entre os coeficientes Mel-Cepstrais o que faria com que a matriz de variância-covariância dos coeficientes deixe de ser diagonal.

Referências

ANDOH, P. Y.; DAVIS, F.; ANTONIO, J. Effects of cutting parameters on acoustic emission signal response during drilling of laminated composites using factorial design method. **Journal of Science and Technology (Ghana)**, v. 27, n. 3, p. 98-106, 10 abr. 2008.

ARAÚJO, A. M. DE L. **Jogos Computacionais Fonoarticulatórios para Crianças com Deficiência Auditiva**. [s.l.] Universidade Estadual de Campinas, 2000.

BHASKARAN, J. *et al.* Monitoring of hard turning using acoustic emission signal. **Journal of Mechanical Science and Technology**, v. 26, n. 2, p. 609-615, 27 abr. 2012.

BIN, G. F. *et al.* Early fault diagnosis of rotating machinery based on wavelet packets - empirical mode decomposition feature extraction and neural network. **Mechanical Systems and Signal Processing**, v. 27, p. 696-711, fev. 2012.

BONIFÁCIO, M. E. .; DINIZ, A. E. Correlating tool wear, tool life, surface roughness and tool vibration in finish turning with coated carbide tools. **Energy**, v. 173, p. 137-144, 1994.

BOUTROS, T.; LIANG, M. Detection and diagnosis of bearing and cutting tool faults using hidden markov models. **Mechanical Systems and Signal Processing**, v. 25, n. 6, p. 2102-2124, ago. 2011.

BOX, G. E. P.; BEHNKEN, D. W. Some new three-level designs for the study of quantitative variables. **Technometrics**, v. 2, p. 455-475, 1960.

BOX, G. E. P.; DRAPER, N. R. **Empirical Model-Building and Response Surfaces, John Wiley & Sons**. 1. ed. [s.l.] John Wiley & Sons, 1987. p. 650

BOX, G. E. P.; HUNTER, W. G.; HUNTER, J. S. **Statistics for Experimenters**. 1. ed. [s.l.] John Wiley & Sons, 1978. p. 690

BROWN, J. C.; SMARAGDIS, P. Hidden markov and gaussian mixture models for automatic call classification. **The Journal of the Acoustical Society of America**, v. 125, n. 6, p. EL221-4, jun. 2009.

CHAUHAN, S. *et al.* A computer-aided MFCC-based HMM system for automatic auscultation. **Computers in Biology and Medicine**, v. 38, n. 2, p. 221-233, 2008.

DAVIS, S.; MERMELSTEIN, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. . 1980, p. 357-366.

DHANALAKSHMI, P.; PALANIVEL, S.; RAMALINGAM, V. Classification of audio signals using SVM and RBFNN. **Expert Systems with Applications**, v. 36, n. 3, p. 6069-6075, abr. 2009.

DHANALAKSHMI, P.; PALANIVEL, S.; RAMALINGAM, V. Classification of audio signals using AANN and GMM. **Applied Soft Computing**, v. 11, n. 1, p. 716-723, jan. 2011.

DHANALAKSHMI, P.; PALANIVEL, S.; RAMALINGAM, V. Pattern classification models for classifying and indexing audio signals. **Engineering Applications of Artificial Intelligence**, v. 24, n. 2, p. 350-357, mar. 2011.

DHANALAKSHMI, P.; PALANIVEL, S.; RAMALINGAM, V. Pattern classification models for classifying and indexing audio signals. **Engineering Applications of Artificial Intelligence**, v. 24, n. 2, p. 350-357, mar. 2011.

DIMLA SR., D. E.; LISTER, P. M. On-line metal cutting tool condition monitoring . I : force and vibration analyses. **International Journal of Machine Tools and Manufacture**, v. 40, p. 739-768, 2000.

ERTUNC, H. M.; LOPARO, K. A.; OCAK, H. Tool wear condition monitoring in drilling operations using hidden markov models (HMMs). **International Journal of Machine Tools and Manufacture**, v. 41, n. 9, p. 1363-1384, jul. 2001.

FADARE, D. . A. *et al.* Influence of cutting parameters and tool wear on acoustic emission signal in high-speed turning of Ti-6Al-4V alloy. **Journal of Emerging Trends in Engineering and Applied Sciences**, v. 3, n. 3, p. 547-555, 2012.

FAHMY, M. M. M. Palmprint recognition based on mel frequency cepstral coefficients feature extraction. **Ain Shams Engineering Journal**, v. 1, n. 1, p. 39-47, set. 2010.

GRABEC, I. *et al.* Monitoring manufacturing processes by utilizing empirical modeling. **Ultrasonics**, v. 36, n. 1-5, p. 263-271, fev. 1998.

HU, Q. *et al.* Fault diagnosis of rotating machinery based on improved wavelet package transform and SVMs ensemble. **Mechanical Systems and Signal Processing**, v. 21, n. 2, p. 688-705, fev. 2007.

JEMIELNIAK, K.; ARRAZOLA, P. J. Application of AE and cutting force signals in tool condition monitoring in micro-milling. **CIRP Journal of Manufacturing Science and Technology**, v. 1, n. 2, p. 97-102, jan. 2008.

JOHSON, R. A.; WICHERM, D. W. **Applied multivariate statistical analysis**. 5. ed. New Jersey: Prentice-Hall Inc, 2002. p. 797

JUNQIN, W.; JUNJUN, Y. **An improved arithmetic of MFCC in speech recognition system.** 2011 International Conference on Electronics Communications and Control ICECC. **Anais**... IEEE, 2011. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6066676>

KAMRUZZAMAN, S. M. *et al.* Speaker identification using MFCC-domain support vector machine. **Computer**, v. I, n. 10, p. 5, 2010.

KASBAN, H. *et al.* Welding defect detection from radiography images with a cepstral approach. **NDT & E International**, v. 44, n. 2, p. 226-231, mar. 2011.

KOOLAGUDI, S. G.; RASTOGI, D.; RAO, K. S. Identification of language using mel-frequency cepstral coefficients (MFCC). **Procedia Engineering**, v. 38, p. 3391-3398, jan. 2012.

KUMARI, R. S. S.; NIDHYANANTHAN, S. S.; G, A. Fused mel feature sets based text-independent speaker identification using gaussian mixture model. **Procedia Engineering**, v. 30, p. 319-326, jan. 2012.

LI, X. A brief review: acoustic emission method for tool wear monitoring during turning. **International Journal of Machine Tools and Manufacture**, v. 42, n. 2, p. 157-165, jan. 2002.

LINDE, Y.; BUZO, A.; GRAY, R. An Algorithm for Vector Quantizer Design. **IEEE Transactions on Communications**, v. 28, n. 1, p. 84-95, 1980.

MARWALA, T.; MAHOLA, U.; NELWAMONDO, F. V. **Hidden markov models and gaussian mixture models for bearing fault detection using fractals.** The 2006 IEEE International Joint Conference on Neural Network Proceedings. **Anais**... IEEE, 2006. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp;jsessionid=HwvyPLxW8JJwYBPLQr5GkT9nwgZXRR0mGL8QgD94Gsw4bR5qQ8PX!399748823?arnumber=1716539contentType=Conference+Publications>. Acesso em: 3 jan. 2013

MAYORGA, P.; BESACIER, L.; LAMY, R. Audio packet loss over IP and speech recognition. **Automatic Speech Recognition**, 2003.

MONTGOMERY, D. C. **Design and analysis of Experiments**. 5. ed. [s.l.] John Wiley & Sons, 2001. p. 699

MONTGOMERY, D. C.; JOHNSON, R. T. Choice of second-order response surface designs for logistic and Poisson regression models. **International Journal of Experimental Design and Process Optimisation**, v. 1, n. 1, p. 2, 2009.

MYERS, R. H.; MONTGOMERY, D. C. **Response Surface Methodology: Process and Product Optimization Using Design of Experiments**. 2. ed. New York: Wiley – Interscience, 1995. p. 700

PAIVA, A.P.; CAMPOS, P.H.; FERREIRA, J.R.; LOPES, L.G.D.; PAIVA, E.J.; BALESTRASSI, P.P.. A multivariate robust parameter design approach for optimization of AISI 52100 hardened steel turning with wiper mixed ceramic tool. **International Journal of Refractory Metals and Hard Materials**, v. 30, n. 1, p. 152-163, jan. 2012.

PICONE, J. W. Signal modeling techniques in speech recognition. **Proceedings of the IEEE**, v. 81, n. 9, p. 1215-1247, 1993.

POLUR, P. D.; MILLER, G. E. Investigation of an HMM/ANN hybrid structure in pattern recognition application using cepstral analysis of dysarthric (distorted) speech signals. **Medical engineering & physics**, v. 28, n. 8, p. 741-8, out. 2006.

RAJA, J. E.; LIM, W. S.; VENKATASESHAIAH, C. Emitted sound analysis for tool flank wear monitoring using hilbert huang transform. **International Journal of Computer and Electrical Engineering**, v. 4, n. 2, 2012.

REYNOLDS, D. A.; QUATIERI, T. F.; DUNN, R. B. Speaker verification using adapted gaussian mixture models. **Digital Signal Processing**, v. 10, n. 1-3, p. 19-41, jan. 2000.

REYNOLDS, D. A.; ROSE, R. C. Robust text-independent speaker identification using gaussian mixture speaker models. **IEEE Transactions On Speech And Audio Processing**, p. 72-83, 1995.

RUBIO, E.; TETI, R. Process monitoring systems for machining using audible sound energy sensors. **Future Manufacturing Systems**, p. 217-235, 2002.

SCHEFFER, C. *et al.* Development of a tool wear-monitoring system for hard turning. **International Journal of Machine Tools and Manufacture**, v. 43, n. 10, p. 973-985, ago. 2003.

SICK, B. On-line and indirect tool wear monitoring in turning with artificial neural networks: a review of more than a decade of research. **Mechanical Systems and Signal Processing**, v. 16, n. 4, p. 487-546, jul. 2002.

VIGNOLO, L. D. *et al.* Evolutionary cepstral coefficients. **Applied Soft Computing**, v. 11, n. 4, p. 3419-3428, jun. 2011.

WANG, C.; MIAO, Z.; MENG, X. **Differential MFCC and vector quantization used for real-time speaker recognition system.** 2008 Congress on Image and Signal Processing. **Anais**... IEEE, 2008. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?ar

number=4566841>

XI, J.; HAN, W.; LIU, Y. Relationship analysis between chaotic characteristics of acoustic emission signal and tool wear condition. **Third International Workshop on Advanced Computational Intelligence**, p. 612-617, ago. 2010.

XIQING, M.; CHUANGWEN, X. Tool wear monitoring of acoustic emission signals from milling processes. **First International Workshop on Education Technology and Computer Science**, p. 431-435, 2009.

YUJIN, Y. Y. Y.; PEIHUA, Z. P. Z.; QUN, Z. Q. Z. **Research of speaker recognition based on combination of LPCC and MFCC.** Intelligent Computing and Intelligent Systems ICIS 2010 IEEE International Conference on. **Anais**...2010

ZHU, K.; WONG, Y. S.; HONG, G. S. Multi-category micro-milling tool wear monitoring with continuous hidden markov models. **Mechanical Systems and Signal Processing**, v. 23, n. 2, p. 547-560, fev. 2009.

ZUNJING, W. U.; ZHIGANG, C. A. O. Improved MFCC-based feature for robust speaker identification. **Tsinghua Science and Technology**, v. 10, n. 2, p. 158-161, 2005.

ANEXO A - TABELAS DE DADOS

Tabela A.1 - Rugosidades médias medidas em cada passo do experimento

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ra1	0,29	0,33	0,28	0,31	0,25	0,28	0,27	0,25	0,26	0,24	0,25	0,26	0,22	0,23	0,25
Ra2	0,51	0,51	0,51	0,50	0,50	0,50	0,51	0,51	0,51	0,50	0,50	0,49	0,49	0,49	0,49
Ra3	0,16	0,16	0,16	0,17	0,17	0,17	0,14	0,14	0,14	0,18	0,18	0,17	0,16	0,16	0,16
Ra4	0,25	0,25	0,25	0,24	0,25	0,25	0,27	0,27	0,27	0,25	0,25	0,25	0,28	0,28	0,28
Ra5	0,19	0,20	0,19	0,18	0,18	0,18	0,19	0,19	0,19	0,18	0,18	0,17	0,22	0,21	0,21
Ra6	0,18	0,18	0,18	0,17	0,17	0,17	0,15	0,15	0,15	0,18	0,18	0,18	0,22	0,23	0,23
Ra7	0,53	0,53	0,52	0,53	0,52	0,53	0,52	0,52	0,51	0,52	0,52	0,52	0,52	0,52	0,52
Ra8	0,15	0,15	0,15	0,14	0,14	0,14	0,15	0,15	0,15	0,13	0,14	0,13	0,16	0,16	0,15
Ra9	0,40	0,40	0,40	0,44	0,43	0,44	0,47	0,47	0,47	0,45	0,45	0,45	0,45	0,45	0,44
Ra10	0,20	0,20	0,20	0,19	0,19	0,19	0,18	0,19	0,18	0,19	0,19	0,19	0,19	0,19	0,18

Tabela A.2 - Dados experimentais do *D-Optimal Design*

	1	2	3	4	5	6	7	8	9	10	11	12	Grupo
Ra1	**0,29**	**0,33**	**0,28**	**0,31**	**0,25**	**0,28**	**0,27**	**0,25**	**0,26**	**0,24**	**0,25**	**0,26**	**1**
Ra2	**0,51**	**0,51**	**0,51**	**0,50**	**0,50**	**0,50**	**0,51**	**0,51**	**0,51**	**0,50**	**0,50**	**0,49**	**2**
Ra3	**0,16**	**0,16**	**0,16**	**0,17**	**0,17**	**0,17**	**0,14**	**0,14**	**0,14**	**0,18**	**0,18**	**0,17**	**3**
Ra4	**0,25**	**0,25**	**0,25**	**0,24**	**0,25**	**0,25**	**0,27**	**0,27**	**0,27**	**0,25**	**0,25**	**0,25**	**1**
Ra5	**0,19**	**0,20**	**0,19**	**0,18**	**0,18**	**0,18**	**0,19**	**0,19**	**0,19**	**0,18**	**0,18**	**0,17**	**4**
Ra6	**0,18**	**0,18**	**0,18**	**0,17**	**0,17**	**0,17**	**0,15**	**0,15**	**0,15**	**0,18**	**0,18**	**0,18**	**3**
Ra7	**0,53**	**0,53**	**0,52**	**0,53**	**0,52**	**0,53**	**0,52**	**0,52**	**0,51**	**0,52**	**0,52**	**0,52**	**2**
Ra8	**0,15**	**0,15**	**0,15**	**0,14**	**0,14**	**0,14**	**0,15**	**0,15**	**0,15**	**0,13**	**0,14**	**0,13**	**3**
Ra9	0,40	0,40	0,40	0,44	0,43	0,44	0,47	0,47	0,47	0,45	0,45	0,45	5
Ra10	**0,20**	**0,20**	**0,20**	**0,19**	**0,19**	**0,19**	**0,18**	**0,19**	**0,18**	**0,19**	**0,19**	**0,19**	**4**

Tabela A.3 – Resultado dos Testes (Perfil de Energia)

			Perfil de Energia		Perfil de Energia	
			4 Gaussinas		8 Gaussinas	
		Teste	Vencedor	P(O|GMM)	Vencedor	P(O|GMM)
Grupo 1	Ra1	vc_186_f_015_ap_015-2	1	1300.396227	1	1301.733090
	Ra1	vc_186_f_015_ap_015-4	1	1127.095108	1	1127.772596
	Ra1	vc_186_f_015_ap_015-6	1	1280.716704	1	1282.102705
	Ra1	vc_186_f_015_ap_015-8	1	1299.984100	1	1301.280878
	Ra1	vc_186_f_015_ap_015-10	1	1303.280961	1	1304.660507
	Ra1	vc_186_f_015_ap_015-12	1	1168.390967	1	1169.139178
	Ra1	vc_186_f_015_ap_015-14	1	1320.896598	1	1322.248233
	Ra4	vc_220_f_015_ap_023-2	1	1099.542795	1	1101.334069
	Ra4	vc_220_f_015_ap_023-4	1	1322.778388	1	1324.092558
	Ra4	vc_220_f_015_ap_023-6	1	1281.633946	1	1282.634273
	Ra4	vc_220_f_015_ap_023-8	1	1334.285741	1	1335.825804
	Ra4	vc_220_f_015_ap_023-10	1	1355.273485	1	1356.657315
	Ra4	vc_220_f_015_ap_023-12	4	688.017899	4	689.275032
	Ra4	vc_220_f_015_ap_023-14	4	401.845900	1	407.216415
			85.71%		85.71%	
Grupo 2	Ra3	vc_240_f_010_ap_010-2	5	1169.285130	5	1243.647075
	Ra3	vc_240_f_010_ap_010-4	5	1205.042072	5	1452.048059
	Ra3	vc_240_f_010_ap_010-6	5	1224.008899	5	1337.493591
	Ra3	vc_240_f_010_ap_010-8	5	1085.013777	2	1048.967343
	Ra3	vc_240_f_010_ap_010-10	2	1041.644722	2	1041.675656
	Ra3	vc_240_f_010_ap_010-12	2	1004.650293	2	1004.749640
	Ra3	vc_240_f_010_ap_010-14	2	1045.558973	2	1045.614910
	Ra6	vc_240_f_010_ap_020-2	1	924.192798	1	924.002913
	Ra6	vc_240_f_010_ap_020-4	5	882.275045	3	872.692352
	Ra6	vc_240_f_010_ap_020-6	5	969.598870	5	1059.139770
	Ra6	vc_240_f_010_ap_020-8	5	950.669969	5	993.822414
	Ra6	vc_240_f_010_ap_020-10	5	977.453676	5	1073.791837
	Ra6	vc_240_f_010_ap_020-12	5	1006.341795	5	1171.652184
	Ra6	vc_240_f_010_ap_020-14	2	578.689471	2	578.666930
	Ra9	vc_200_f_010_ap_010-2	5	1053.034769	2	1050.347579
	Ra9	vc_200_f_010_ap_010-4	2	839.987272	2	840.060103
	Ra9	vc_200_f_010_ap_010-6	2	424.644145	2	424.686024
	Ra9	vc_200_f_010_ap_010-8	2	675.101435	2	675.141215
	Ra9	vc_200_f_010_ap_010-10	2	473.997017	2	474.059465
	Ra9	vc_200_f_010_ap_010-12	2	750.588409	2	750.646120
	Ra9	vc_200_f_010_ap_010-14	2	383.367887	2	383.432531
			47.62%		57.14%	

Tabela A.4 – Resultados dos Testes (Perfil de Energia)

			Perfil de Energia		Perfil de Energia	
			4 Gaussinas		8 Gaussinas	
		Teste	Vencedor	P(O|GMM)	Vencedor	P(O|GMM)
Grupo 3	Ra5	vc_240_f_015_ap_015-2	1	1321.324364	1	1322.403684
	Ra5	vc_240_f_015_ap_015-4	3	1013.256975	3	1013.501642
	Ra5	vc_240_f_015_ap_015-6	1	1110.859751	1	1110.958636
	Ra5	vc_240_f_015_ap_015-8	1	1060.963225	1	1061.137601
	Ra5	vc_240_f_015_ap_015-10	5	1111.792230	5	1095.164629
	Ra5	vc_240_f_015_ap_015-12	5	977.076342	2	958.874213
	Ra5	vc_240_f_015_ap_015-14	1	1317.298296	1	1318.541114
	Ra11	vc_200_f_010_ap_020-2	5	1280.002979	5	1449.810135
	Ra11	vc_200_f_010_ap_020-4	2	979.976265	2	980.061939
	Ra11	vc_200_f_010_ap_020-6	2	1048.094940	2	1048.153781
	Ra11	vc_200_f_010_ap_020-8	2	1002.258676	2	1002.284535
	Ra11	vc_200_f_010_ap_020-10	2	1061.568938	2	1061.563389
	Ra11	vc_200_f_010_ap_020-12	5	1178.451693	2	1041.475964
	Ra11	vc_200_f_010_ap_020-14	2	1068.946351	2	1068.968609
			7.14%		7.14%	
Grupo 4	Ra2	vc_220_f_023_ap_015-2	1	577.897689	1	578.458635
	Ra2	vc_220_f_023_ap_015-4	1	622.145117	1	622.835425
	Ra2	vc_220_f_023_ap_015-6	1	452.747852	1	453.401878
	Ra2	vc_220_f_023_ap_015-8	4	341.461923	4	341.533175
	Ra2	vc_220_f_023_ap_015-10	4	386.207835	4	386.282552
	Ra2	vc_220_f_023_ap_015-12	4	401.871780	4	401.948499
	Ra2	vc_220_f_023_ap_015-14	1	450.420771	1	451.007613
	Ra8	vc_240_f_020_ap_020-2	1	827.500684	1	827.020021
	Ra8	vc_240_f_020_ap_020-4	1	790.006165	1	789.500365
	Ra8	vc_240_f_020_ap_020-6	4	762.314490	4	762.386524
	Ra8	vc_240_f_020_ap_020-8	4	750.393483	4	750.465543
	Ra8	vc_240_f_020_ap_020-10	2	590.546251	2	590.610776
	Ra8	vc_240_f_020_ap_020-12	4	745.673126	4	745.694806
	Ra8	vc_240_f_020_ap_020-14	3	726.573657	3	726.697105
			42.86%		42.86%	
Grupo 5	Ra10	vc_200_f_020_ap_010-2	5	946.435885	5	1068.603614
	Ra10	vc_200_f_020_ap_010-4	5	962.849789	5	1118.111174
	Ra10	vc_200_f_020_ap_010-6	5	971.069985	5	1082.203543
	Ra10	vc_200_f_020_ap_010-8	3	855.346351	3	855.557886
	Ra10	vc_200_f_020_ap_010-10	5	990.690903	5	1189.852305
	Ra10	vc_200_f_020_ap_010-12	2	749.013977	2	749.026271
	Ra10	vc_200_f_020_ap_010-14	5	1027.218845	5	1199.107348
			71.43%		71.43%	

Tabela A.5 – Resultado dos Testes (MFCC).

			Mel Cepstrais 4 Gaussianas		Mel Cepstrais 8 Gaussianas	
		Teste	Vencedor	P(O|GMM)	Vencedor	P(O|GMM)
Grupo 1	Ra1	vc_186_f_015_ap_015-2	1	-13463.0581	1	-13453.7966
	Ra1	vc_186_f_015_ap_015-4	1	-13575.2433	1	-13496.7129
	Ra1	vc_186_f_015_ap_015-6	1	-13705.2797	1	-13496.7129
	Ra1	vc_186_f_015_ap_015-8	1	-14189.2863	1	-14192.5132
	Ra1	vc_186_f_015_ap_015-10	1	-13871.0881	1	-13724.0570
	Ra1	vc_186_f_015_ap_015-12	1	-14422.0803	1	-13724.0570
	Ra1	vc_186_f_015_ap_015-14	1	-13877.5949	1	-13692.9067
	Ra4	vc_220_f_015_ap_023-2	1	-14106.2085	1	-13884.0255
	Ra4	vc_220_f_015_ap_023-4	1	-13812.9316	1	-13601.5999
	Ra4	vc_220_f_015_ap_023-6	1	-13999.1772	1	-13481.0632
	Ra4	vc_220_f_015_ap_023-8	1	-13662.4256	1	-13528.7005
	Ra4	vc_220_f_015_ap_023-10	1	-13704.8891	1	-13452.6827
	Ra4	vc_220_f_015_ap_023-12	1	-14323.7615	1	-14217.4960
	Ra4	vc_220_f_015_ap_023-14	1	-14377.0524	1	-14168.0461
			100.00%		100.00%	
Grupo 2	Ra3	vc_240_f_010_ap_010-2	2	-13873.8360	2	-13869.6283
	Ra3	vc_240_f_010_ap_010-4	2	-14205.7835	2	-14194.6519
	Ra3	vc_240_f_010_ap_010-6	2	-13656.4553	2	-13664.5636
	Ra3	vc_240_f_010_ap_010-8	2	-13709.1491	2	-13673.0018
	Ra3	vc_240_f_010_ap_010-10	2	-13815.2016	2	-13804.7328
	Ra3	vc_240_f_010_ap_010-12	2	-14089.5948	2	-13850.5639
	Ra3	vc_240_f_010_ap_010-14	2	-13723.4649	2	-13533.6005
	Ra6	vc_240_f_010_ap_020-2	2	-11305.0511	2	-11056.5649
	Ra6	vc_240_f_010_ap_020-4	2	-10743.6051	2	-10697.1446
	Ra6	vc_240_f_010_ap_020-6	2	-10823.8350	2	-10699.6183
	Ra6	vc_240_f_010_ap_020-8	2	-10791.5142	2	-10714.0710
	Ra6	vc_240_f_010_ap_020-10	2	-11178.6626	2	-11012.1732
	Ra6	vc_240_f_010_ap_020-12	2	-11130.3379	2	-10898.8204
	Ra6	vc_240_f_010_ap_020-14	2	-11527.0246	2	-11521.2382
	Ra9	vc_200_f_010_ap_010-2	2	-15029.2335	2	-15085.4798
	Ra9	vc_200_f_010_ap_010-4	2	-14467.2201	2	-14004.2804
	Ra9	vc_200_f_010_ap_010-6	2	-14720.5738	2	-14223.8849
	Ra9	vc_200_f_010_ap_010-8	2	-13727.3003	2	-13729.3926
	Ra9	vc_200_f_010_ap_010-10	2	-13727.3003	2	-13906.4723
	Ra9	vc_200_f_010_ap_010-12	2	-14132.5322	2	-13851.2358
	Ra9	vc_200_f_010_ap_010-14	2	-14282.7366	2	-14026.1711
			100.00%		100.00%	

Tabela A.6 – Resultados dos Testes (MFCC).

			Mel Cepstrais 4 Gaussianas		Mel Cepstrais 8 Gaussianas	
		Teste	Vencedor	P(O|GMM)	Vencedor	P(O|GMM)
Groupo 3	Ra5	vc_240_f_015_ap_015-2	3	-13617.4694	3	-13517.7563
	Ra5	vc_240_f_015_ap_015-4	3	-13672.7506	3	-13755.6048
	Ra5	vc_240_f_015_ap_015-6	3	-13461.6585	3	-13566.6122
	Ra5	vc_240_f_015_ap_015-8	3	-13451.8101	3	-13354.5046
	Ra5	vc_240_f_015_ap_015-10	3	-13950.4080	3	-13556.8276
	Ra5	vc_240_f_015_ap_015-12	3	-15832.7186	3	-15122.6559
	Ra5	vc_240_f_015_ap_015-14	3	-13695.0441	3	-13598.0826
	Ra11	vc_200_f_010_ap_020-2	3	-14693.9539	3	-14680.0026
	Ra11	vc_200_f_010_ap_020-4	5	-14302.8819	3	-14003.4788
	Ra11	vc_200_f_010_ap_020-6	3	-14063.9784	3	-13804.9308
	Ra11	vc_200_f_010_ap_020-8	3	-14827.6478	3	-14632.6341
	Ra11	vc_200_f_010_ap_020-10	3	-14072.0950	3	-13903.5204
	Ra11	vc_200_f_010_ap_020-12	3	-13762.8207	3	-13459.1646
	Ra11	vc_200_f_010_ap_020-14	3	-13906.4130	3	-13715.3053
			92.86%		100.00%	
Groupo 4	Ra2	vc_220_f_023_ap_015-2	3	-7123.9096	3	-7264.0439
	Ra2	vc_220_f_023_ap_015-4	1	-7356.2066	1	-7208.6740
	Ra2	vc_220_f_023_ap_015-6	1	-7112.2838	1	-7109.3537
	Ra2	vc_220_f_023_ap_015-8	4	-6693.4941	4	-6687.4055
	Ra2	vc_220_f_023_ap_015-10	4	-6815.7354	4	-6910.2807
	Ra2	vc_220_f_023_ap_015-12	4	-6824.7435	4	-6799.4180
	Ra2	vc_220_f_023_ap_015-14	4	-7100.0741	4	-7029.6875
	Ra8	vc_240_f_020_ap_020-2	4	-9869.1702	4	-9890.4305
	Ra8	vc_240_f_020_ap_020-4	4	-9322.2844	4	-9123.5867
	Ra8	vc_240_f_020_ap_020-6	4	-9382.3254	4	-9265.3969
	Ra8	vc_240_f_020_ap_020-8	4	-9293.1901	4	-9202.7250
	Ra8	vc_240_f_020_ap_020-10	2	-11227.0100	2	-11171.5973
	Ra8	vc_240_f_020_ap_020-12	4	-9429.8508	4	-9296.1907
	Ra8	vc_240_f_020_ap_020-14	4	-9271.5701	4	-9232.0805
			71.43%		71.43%	
Grupo 5	Ra10	vc_200_f_020_ap_010-2	5	-10515.7713	5	-10472.1596
	Ra10	vc_200_f_020_ap_010-4	5	-10333.6820	5	-10262.3065
	Ra10	vc_200_f_020_ap_010-6	5	-10597.5681	5	-10501.6553
	Ra10	vc_200_f_020_ap_010-8	5	-11433.8622	5	-11343.6548
	Ra10	vc_200_f_020_ap_010-10	5	-10288.6407	5	-10195.3253
	Ra10	vc_200_f_020_ap_010-12	5	-10515.9049	5	-10432.6145
	Ra10	vc_200_f_020_ap_010-14	5	-10875.9120	5	-10835.7117
			100.00%		100.00%	

Tabela A.7 – *D-Optimal Design* de P(O|GMM) (1ª., 2ª. e 3ª. réplicas).

Vc	f	ap	PE4G[1]	PE8G[1]	MFCC4G[1]	MFCC8G[1]
200.00	0.10	0.10	1053.0348	1050.3476	-15029.2335	-15085.4798
240.00	0.10	0.10	1169.2851	1243.6471	-13873.8360	-13869.6283
200.00	0.20	0.10	946.4359	1068.6036	-10515.7713	-10472.1596
240.00	0.20	0.10	1321.3244	1322.4037	-13617.4694	-13517.7563
200.00	0.10	0.20	1280.0030	1449.8101	-14693.9539	-14680.0026
240.00	0.10	0.20	924.1928	924.0029	-11305.0511	-11056.5649
200.00	0.20	0.20	*	*	*	*
240.00	0.20	0.20	827.5007	827.0200	-9869.1702	-9890.4305
186.36	0.15	0.15	1300.3962	1301.7331	-13463.0581	-13453.7966
253.64	0.15	0.15	*	*	*	*
220.00	0.07	0.15	*	*	*	*
220.00	0.23	0.15	577.8977	578.4586	-7123.9096	-7264.0439
220.00	0.15	0.07	*	*	*	*
220.00	0.15	0.23	1099.5428	1101.3341	-14106.2085	-13884.0255
200.00	0.10	0.10	839.9873	840.0601	-14467.2201	-14004.2804
240.00	0.10	0.10	1205.0421	1452.0481	-14205.7835	-14194.6519
200.00	0.20	0.10	962.8498	1118.1112	-10333.6820	-10262.3065
240.00	0.20	0.10	1013.2570	1013.5016	-13672.7506	-13755.6048
200.00	0.10	0.20	979.9763	980.0619	-14302.8819	-14003.4788
240.00	0.10	0.20	882.2750	872.6924	-10743.6051	-10697.1446
200.00	0.20	0.20	*	*	*	*
240.00	0.20	0.20	790.0062	789.5004	-9322.2844	-9123.5867
186.36	0.15	0.15	1127.0951	1127.7726	-13575.2433	-13496.7129
253.64	0.15	0.15	*	*	*	*
220.00	0.07	0.15	*	*	*	*
220.00	0.23	0.15	622.1451	622.8354	-7356.2066	-7208.6740
220.00	0.15	0.07	*	*	*	*
220.00	0.15	0.23	1322.7784	1324.0926	-13812.9316	-13601.5999
200.00	0.10	0.10	424.6441	424.6860	-14720.5738	-14223.8849
240.00	0.10	0.10	1224.0089	1337.4936	-13656.4553	-13664.5636
200.00	0.20	0.10	971.0700	1082.2035	-10597.5681	-10501.6553
240.00	0.20	0.10	1110.8598	1110.9586	-13461.6585	-13566.6122
200.00	0.10	0.20	1048.0949	1048.1538	-14063.9784	-13804.9308
240.00	0.10	0.20	969.5989	1059.1398	-10823.8350	-10699.6183
200.00	0.20	0.20	*	*	*	*
240.00	0.20	0.20	762.3145	762.3865	-9382.3254	-9265.3969
186.36	0.15	0.15	1280.7167	1282.1027	-13705.2797	-13496.7129
253.64	0.15	0.15	*	*	*	*
220.00	0.07	0.15	*	*	*	*
220.00	0.23	0.15	452.7479	453.4019	-7112.2838	-7109.3537
220.00	0.15	0.07	*	*	*	*
220.00	0.15	0.23	1281.6339	1282.6343	-13999.1772	-13481.0632

1 – Os símbolos (*) representam experimentos excluídos do arranjo CCD pelo critério *D-Optimal*.

Tabela A.8 – *D-Optimal Design* de P(O|GMM) (4ª., 5ª. E 6ª. réplicas).

Vc	f	ap	PE4G[1]	PE8G[1]	MFCC4G[1]	MFCC8G[1]
200.00	0.10	0.10	675.1014	675.1412	-13727.3003	-13729.3926
240.00	0.10	0.10	1085.0138	1048.9673	-13709.1491	-13673.0018
200.00	0.20	0.10	855.3464	855.5579	-11433.8622	-11343.6548
240.00	0.20	0.10	1060.9632	1061.1376	-13451.8101	-13354.5046
200.00	0.10	0.20	1002.2587	1002.2845	-14827.6478	-14632.6341
240.00	0.10	0.20	950.6700	993.8224	-10791.5142	-10714.0710
200.00	0.20	0.20	*	*	*	*
240.00	0.20	0.20	750.3935	750.4655	-9293.1901	-9202.7250
186.36	0.15	0.15	1299.9841	1301.2809	-14189.2863	-14192.5132
253.64	0.15	0.15	*	*	*	*
220.00	0.07	0.15	*	*	*	*
220.00	0.23	0.15	341.4619	341.5332	-6693.4941	-6687.4055
220.00	0.15	0.07	*	*	*	*
220.00	0.15	0.23	1334.2857	1335.8258	-13662.4256	-13528.7005
200.00	0.10	0.10	473.9970	474.0595	-13727.3003	-13906.4723
240.00	0.10	0.10	1041.6447	1041.6757	-13815.2016	-13804.7328
200.00	0.20	0.10	990.6909	1189.8523	-10288.6407	-10195.3253
240.00	0.20	0.10	1111.7922	1095.1646	-13950.4080	-13556.8276
200.00	0.10	0.20	1061.5689	1061.5634	-14072.0950	-13903.5204
240.00	0.10	0.20	977.4537	1073.7918	-11178.6626	-11012.1732
200.00	0.20	0.20	*	*	*	*
240.00	0.20	0.20	590.5463	590.6108	-11227.0100	-11171.5973
186.36	0.15	0.15	1303.2810	1304.6605	-13871.0881	-13724.0570
253.64	0.15	0.15	*	*	*	*
220.00	0.07	0.15	*	*	*	*
220.00	0.23	0.15	386.2078	386.2826	-6815.7354	-6910.2807
220.00	0.15	0.07	*	*	*	*
220.00	0.15	0.23	1355.2735	1356.6573	-13704.8891	-13452.6827
200.00	0.10	0.10	750.5884	750.6461	-14132.5322	-13851.2358
240.00	0.10	0.10	1004.6503	1004.7496	-14089.5948	-13850.5639
200.00	0.20	0.10	749.0140	749.0263	-10515.9049	-10432.6145
240.00	0.20	0.10	977.0763	958.8742	-15832.7186	-15122.6559
200.00	0.10	0.20	1178.4517	1041.4760	-13762.8207	-13459.1646
240.00	0.10	0.20	1006.3418	1171.6522	-11130.3379	-10898.8204
200.00	0.20	0.20	*	*	*	*
240.00	0.20	0.20	745.6731	745.6948	-9429.8508	-9296.1907
186.36	0.15	0.15	1168.3910	1169.1392	-14422.0803	-13724.0570
253.64	0.15	0.15	*	*	*	*
220.00	0.07	0.15	*	*	*	*
220.00	0.23	0.15	401.8718	401.9485	-6824.7435	-6799.4180
220.00	0.15	0.07	*	*	*	*
220.00	0.15	0.23	688.0179	689.2750	-14323.7615	-14217.4960

[1] – Os símbolos (*) representam experimentos excluídos do arranjo CCD pelo critério *D-Optimal*.

Tabela A.9 - *D-Optimal Design* de P(O|GMM) (7ª. réplica).

Vc	f	ap	PE4G[1]	PE8G[1]	MFCC4G[1]	MFCC8G[1]
200.00	0.10	0.10	383.3679	383.4325	-14282.7366	-14026.1711
240.00	0.10	0.10	1045.5590	1045.6149	-13723.4649	-13533.6005
200.00	0.20	0.10	1027.2188	1199.1073	-10875.9120	-10835.7117
240.00	0.20	0.10	1317.2983	1318.5411	-13695.0441	-13598.0826
200.00	0.10	0.20	1068.9464	1068.9686	-13906.4130	-13715.3053
240.00	0.10	0.20	578.6895	578.6669	-11527.0246	-11521.2382
200.00	0.20	0.20	*	*	*	*
240.00	0.20	0.20	726.5737	726.6971	-9271.5701	-9232.0805
186.36	0.15	0.15	1320.8966	1322.2482	-13877.5949	-13692.9067
253.64	0.15	0.15	*	*	*	*
220.00	0.07	0.15	*	*	*	*
220.00	0.23	0.15	450.4208	451.0076	-7100.0741	-7029.6875
220.00	0.15	0.07	*	*	*	*
220.00	0.15	0.23	401.8459	407.2164	-14377.0524	-14168.0461

1 – Os símbolos (*) representam experimentos excluídos do arranjo CCD pelo critério *D-Optimal*.

Tabela A.10 – *D-Optimal Design* para Rugosidades (1ª., 2ª. e 3ª. réplicas).

Vc	f	ap	Ra	Rq	Rt	Ry	Rz	Rsm
200.00	0.10	0.10	0.153	0.188	1.444	1.032	0.706	111.167
240.00	0.10	0.10	0.250	0.317	2.576	1.862	1.245	67.833
200.00	0.20	0.10	0.403	0.491	3.541	2.408	1.603	189.250
240.00	0.20	0.10	0.480	0.572	2.901	2.425	1.879	141.417
200.00	0.10	0.20	0.199	0.252	2.177	1.488	1.002	65.917
240.00	0.10	0.20	0.195	0.243	2.003	1.431	0.932	111.417
200.00	0.20	0.20	*	*	*	*	*	*
240.00	0.20	0.20	0.528	0.638	3.482	2.978	2.270	194.917
186.36	0.15	0.15	0.326	0.425	3.635	2.376	1.531	141.333
253.64	0.15	0.15	*	*	*	*	*	*
220.00	0.07	0.15	*	*	*	*	*	*
220.00	0.23	0.15	0.513	0.615	3.348	2.728	2.121	220.750
220.00	0.15	0.07	*	*	*	*	*	*
220.00	0.15	0.23	0.162	0.203	1.513	1.076	0.720	116.333
200.00	0.10	0.10	0.138	0.178	1.461	1.052	0.760	79.000
240.00	0.10	0.10	0.243	0.310	2.688	1.821	1.213	73.750
200.00	0.20	0.10	0.436	0.512	2.766	2.227	1.662	196.083
240.00	0.20	0.10	0.477	0.572	3.258	2.484	1.850	123.083
200.00	0.10	0.20	0.189	0.233	1.744	1.301	0.983	49.917
240.00	0.10	0.20	0.180	0.227	2.103	1.293	0.919	79.583
200.00	0.20	0.20	*	*	*	*	*	*
240.00	0.20	0.20	0.525	0.629	3.238	2.838	2.238	195.500
186.36	0.15	0.15	0.308	0.397	3.080	2.148	1.414	105.417
253.64	0.15	0.15	*	*	*	*	*	*
220.00	0.07	0.15	*	*	*	*	*	*
220.00	0.23	0.15	0.504	0.613	3.242	2.761	2.222	224.333
220.00	0.15	0.07	*	*	*	*	*	*
220.00	0.15	0.23	0.172	0.220	1.877	1.328	0.852	93.417
200.00	0.10	0.10	0.140	0.178	1.535	1.048	0.752	77.167
240.00	0.10	0.10	0.246	0.318	2.908	1.933	1.247	77.083
200.00	0.20	0.10	0.435	0.509	2.738	2.183	1.647	193.000
240.00	0.20	0.10	0.474	0.573	3.323	2.523	1.853	122.750
200.00	0.10	0.20	0.188	0.233	1.656	1.293	0.988	48.667
240.00	0.10	0.20	0.178	0.225	2.127	1.279	0.897	72.833
200.00	0.20	0.20	*	*	*	*	*	*
240.00	0.20	0.20	0.526	0.630	3.230	2.840	2.249	195.917
186.36	0.15	0.15	0.275	0.341	2.363	1.914	1.283	80.917
253.64	0.15	0.15	*	*	*	*	*	*
220.00	0.07	0.15	*	*	*	*	*	*
220.00	0.23	0.15	0.504	0.613	3.269	2.735	2.218	220.667
220.00	0.15	0.07	*	*	*	*	*	*
220.00	0.15	0.23	0.171	0.220	1.879	1.331	0.858	87.667

1 – Os símbolos (*) representam experimentos excluídos do arranjo CCD pelo critério *D-Optimal*.

Tabela A.11 – *D-Optimal Design* para as Rugosidades (4ª., 5ª. E 6ª. réplicas).

Vc	f	ap	Ra	Rq	Rt	Ry	Rz	Rsm
200.00	0.10	0.10	0.148	0.187	1.547	1.111	0.767	95.333
240.00	0.10	0.10	0.266	0.327	2.403	1.773	1.240	78.083
200.00	0.20	0.10	0.469	0.558	3.032	2.483	1.892	190.333
240.00	0.20	0.10	0.453	0.554	3.158	2.455	1.783	104.833
200.00	0.10	0.20	0.186	0.233	1.805	1.317	0.947	69.500
240.00	0.10	0.20	0.192	0.249	2.175	1.507	1.028	53.083
200.00	0.20	0.20	*	*	*	*	*	*
240.00	0.20	0.20	0.516	0.623	3.540	2.940	2.208	193.250
186.36	0.15	0.15	0.253	0.308	2.125	1.586	1.065	139.917
253.64	0.15	0.15	*	*	*	*	*	*
220.00	0.07	0.15	*	*	*	*	*	*
220.00	0.23	0.15	0.512	0.598	2.918	2.554	2.010	209.917
220.00	0.15	0.07	*	*	*	*	*	*
220.00	0.15	0.23	0.142	0.178	1.286	1.023	0.743	78.583
200.00	0.10	0.10	0.134	0.171	1.306	0.993	0.734	82.750
240.00	0.10	0.10	0.254	0.316	2.467	1.775	1.288	71.000
200.00	0.20	0.10	0.451	0.543	3.250	2.485	1.893	164.833
240.00	0.20	0.10	0.348	0.429	2.451	1.941	1.464	105.750
200.00	0.10	0.20	0.187	0.232	1.612	1.277	0.943	65.417
240.00	0.10	0.20	0.180	0.241	2.374	1.503	1.041	69.250
200.00	0.20	0.20	*	*	*	*	*	*
240.00	0.20	0.20	0.520	0.628	3.251	2.813	2.220	198.833
186.36	0.15	0.15	0.240	0.319	3.493	1.772	1.269	119.583
253.64	0.15	0.15	*	*	*	*	*	*
220.00	0.07	0.15	*	*	*	*	*	*
220.00	0.23	0.15	0.498	0.586	3.041	2.605	2.052	208.083
220.00	0.15	0.07	*	*	*	*	*	*
220.00	0.15	0.23	0.178	0.220	1.612	1.268	0.893	85.917
200.00	0.10	0.10	0.134	0.167	1.299	0.980	0.733	78.167
240.00	0.10	0.10	0.251	0.316	2.626	1.825	1.296	70.667
200.00	0.20	0.10	0.453	0.543	3.213	2.467	1.875	166.000
240.00	0.20	0.10	0.348	0.428	2.487	1.957	1.468	104.250
200.00	0.10	0.20	0.185	0.232	1.678	1.282	0.934	69.250
240.00	0.10	0.20	0.173	0.230	2.296	1.457	1.021	62.500
200.00	0.20	0.20	*	*	*	*	*	*
240.00	0.20	0.20	0.521	0.627	3.331	2.837	2.228	198.917
186.36	0.15	0.15	0.258	0.328	2.876	1.898	1.219	85.917
253.64	0.15	0.15	*	*	*	*	*	*
220.00	0.07	0.15	*	*	*	*	*	*
220.00	0.23	0.15	0.494	0.583	2.995	2.582	2.018	212.333
220.00	0.15	0.07	*	*	*	*	*	*
220.00	0.15	0.23	0.173	0.214	1.525	1.207	0.875	88.500

1 – Os símbolos (*) representam experimentos excluídos do arranjo CCD pelo critério *D-Optimal*.

Tabela A.12 – *D-Optimal Design* para as Rugosidades (7ª. réplica).

Vc	f	ap	Ra	Rq	Rt	Ry	Rz	Rsm
200.00	0.10	0.10	*	0.194	1.432	1.087	0.813	68.750
240.00	0.10	0.10	*	0.357	2.953	2.118	1.414	72.083
200.00	0.20	0.10	*	0.533	2.914	2.412	1.791	107.750
240.00	0.20	0.10	*	*	*	*	*	*
200.00	0.10	0.20	*	0.234	1.791	1.450	1.152	45.167
240.00	0.10	0.20	*	0.258	1.889	1.399	0.921	136.083
200.00	0.20	0.20	*	*	*	*	*	*
240.00	0.20	0.20	*	0.625	3.590	2.935	2.240	191.833
186.36	0.15	0.15	*	0.286	1.993	1.431	1.093	72.333
253.64	0.15	0.15	*	*	*	*	*	*
220.00	0.07	0.15	*	*	*	*	*	*
220.00	0.23	0.15	*	0.575	2.947	2.488	1.900	204.000
220.00	0.15	0.07	*	*	*	*	*	*
220.00	0.15	0.23	*	0.198	1.522	1.166	0.862	74.417

1 – Os símbolos (*) representam experimentos excluídos do arranjo CCD pelo critério *D-Optimal*.

Tabela A.13 – Medições de Rugosidade.

	Rug.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Média	Var
1	Ra	0.291	0.326	0.281	0.308	0.249	0.275	0.270	0.253	0.258	0.240	0.252	0.258				0.272	0.0007
	Ry	2.031	2.376	2.235	2.148	1.683	1.914	2.045	1.586	1.898	1.772	1.841	1.898	1.513	1.431	1.707	1.872	0.0714
	Rz	1.327	1.531	1.389	1.414	1.179	1.283	1.338	1.065	1.188	1.269	1.184	1.219	1.049	1.093	1.255	1.252	0.0181
	Rq	0.366	0.425	0.366	0.397	0.310	0.341	0.343	0.308	0.325	0.319	0.317	0.328	0.271	0.286	0.317	0.334	0.0016
	Rt	2.679	3.635	3.105	3.080	2.309	2.363	2.793	2.125	2.645	3.493	2.826	2.876	2.172	1.993	2.548	2.709	0.2333
	Rsm	160.833	141.333	135.333	105.417	92.667	80.917	75.250	139.917	123.417	119.583	92.167	85.917	76.083	72.333	63.500	104.311	920.9459
2	Ra	0.512	0.513	0.511	0.504	0.503	0.504	0.513	0.512	0.513	0.498	0.498	0.494				0.506	0.0000
	Ry	2.757	2.728	2.704	2.761	2.746	2.735	2.562	2.554	2.549	2.605	2.593	2.582	2.515	2.488	2.490	2.625	0.0105
	Rz	2.147	2.121	2.126	2.222	2.215	2.218	2.020	2.010	2.023	2.052	2.011	2.018	1.918	1.900	1.893	2.060	0.0124
	Rq	0.617	0.615	0.613	0.613	0.611	0.613	0.597	0.598	0.599	0.586	0.585	0.583	0.580	0.575	0.574	0.597	0.0002
	Rt	3.389	3.348	3.247	3.242	3.256	3.269	2.903	2.918	2.923	3.041	3.004	2.995	3.013	2.947	2.936	3.095	0.0303
	Rsm	219.250	220.750	220.167	224.333	219.000	220.667	211.917	209.917	211.917	208.083	207.583	212.333	196.583	204.000	198.667	212.344	70.6447
3	Ra	0.160	0.162	0.162	0.172	0.173	0.171	0.143	0.142	0.141	0.178	0.177	0.173				0.163	0.0002
	Ry	1.057	1.076	1.083	1.328	1.323	1.331	1.029	1.023	1.037	1.268	1.231	1.207	1.233	1.166	1.176	1.171	0.0130
	Rz	0.708	0.720	0.721	0.852	0.865	0.858	0.748	0.743	0.744	0.893	0.878	0.875	0.890	0.862	0.863	0.815	0.0053
	Rq	0.200	0.203	0.203	0.220	0.219	0.220	0.178	0.178	0.176	0.220	0.218	0.214	0.202	0.198	0.196	0.203	0.0003
	Rt	1.412	1.513	1.508	1.877	1.813	1.879	1.343	1.286	1.344	1.612	1.588	1.525	1.647	1.522	1.488	1.557	0.0343
	Rsm	129.500	116.333	129.000	93.417	92.500	87.667	81.333	78.583	76.000	85.917	84.083	88.500	78.667	74.417	73.583	91.300	347.7776
4	Ra	0.253	0.250	0.251	0.243	0.249	0.246	0.270	0.266	0.266	0.254	0.254	0.251				0.254	0.0001
	Ry	1.963	1.862	1.883	1.821	1.883	1.933	1.844	1.773	1.796	1.775	1.823	1.825	2.143	2.118	2.115	1.904	0.0159
	Rz	1.285	1.245	1.248	1.213	1.230	1.247	1.268	1.240	1.240	1.288	1.295	1.296	1.421	1.414	1.407	1.289	0.0048
	Rq	0.321	0.317	0.318	0.310	0.318	0.318	0.334	0.327	0.329	0.316	0.318	0.316	0.359	0.357	0.356	0.328	0.0003
	Rt	2.849	2.576	2.648	2.688	2.918	2.908	2.526	2.403	2.498	2.467	2.668	2.626	2.897	2.953	2.898	2.701	0.0353
	Rsm	67.500	67.833	69.667	73.750	76.500	77.083	79.583	78.083	78.833	71.000	70.917	70.667	71.333	72.083	70.833	73.044	15.9155
5	Ra	0.194	0.195	0.188	0.180	0.180	0.178	0.191	0.192	0.191	0.180	0.178	0.173				0.185	0.0001
	Ry	1.457	1.431	1.412	1.293	1.244	1.279	1.495	1.507	1.504	1.503	1.526	1.457	1.431	1.399	1.383	1.421	0.0079
	Rz	0.943	0.932	0.911	0.919	0.908	0.897	1.020	1.028	1.023	1.041	1.052	1.021	0.946	0.921	0.917	0.965	0.0033
	Rq	0.243	0.243	0.238	0.227	0.223	0.225	0.247	0.249	0.247	0.241	0.235	0.230	0.267	0.258	0.253	0.242	0.0002
	Rt	2.043	2.003	1.984	2.103	1.909	2.127	2.138	2.175	2.216	2.374	2.425	2.296	1.934	1.889	1.891	2.100	0.0299
	Rsm	120.333	111.417	112.750	79.583	80.667	72.833	53.833	53.083	53.750	69.250	67.583	62.500	125.000	136.083	123.417	88.139	891.0215
6	Ra	0.180	0.177	0.178	0.172	0.172	0.170	0.148	0.148	0.147	0.179	0.176	0.176				0.168	0.0002
	Ry	1.212	1.170	1.190	1.341	1.318	1.303	1.039	1.106	1.047	1.358	1.321	1.322	1.639	1.704	1.672	1.316	0.0447
	Rz	0.783	0.763	0.768	0.846	0.843	0.838	0.753	0.763	0.754	0.949	0.938	0.928	1.123	1.139	1.118	0.887	0.0198
	Rq	0.223	0.218	0.219	0.223	0.222	0.219	0.184	0.186	0.182	0.228	0.223	0.223	0.288	0.291	0.287	0.228	0.0012
	Rt	1.679	1.608	1.633	2.035	1.951	1.978	1.324	1.625	1.368	2.058	1.988	2.010	2.248	2.503	2.363	1.891	0.1189
	Rsm	152.833	163.833	162.583	107.083	99.750	105.333	82.750	82.000	83.917	74.333	73.000	73.667	81.667	83.167	84.083	100.667	1049.2153
7	Ra	0.483	0.480	0.481	0.477	0.473	0.474	0.453	0.453	0.452	0.348	0.346	0.348				0.439	0.0032
	Ry	2.436	2.425	2.409	2.484	2.537	2.523	2.393	2.455	2.392	1.941	1.945	1.957				2.325	0.0539
	Rz	1.893	1.879	1.876	1.850	1.853	1.853	1.773	1.783	1.757	1.464	1.458	1.468				1.742	0.0302
	Rq	0.573	0.572	0.571	0.572	0.573	0.573	0.547	0.554	0.546	0.429	0.429	0.428				0.530	0.0039
	Rt	2.755	2.901	2.893	3.258	3.298	3.323	3.058	3.158	3.043	2.451	18.779	2.487				4.284	20.9224
	Rsm	138.417	141.417	143.417	123.083	121.750	122.750	108.500	104.833	112.750	105.750	105.167	104.250				119.340	222.4627
8	Ra	0.526	0.528	0.524	0.525	0.523	0.526	0.516	0.516	0.512	0.520	0.523	0.521				0.522	0.0000
	Ry	2.969	2.978	2.977	2.838	2.826	2.840	2.953	2.940	2.890	2.813	2.828	2.837	2.924	2.935	2.936	2.899	0.0039
	Rz	2.275	2.270	2.263	2.238	2.236	2.249	2.235	2.208	2.190	2.220	2.220	2.228	2.244	2.240	2.226	2.236	0.0005
	Rq	0.640	0.638	0.637	0.629	0.628	0.630	0.622	0.623	0.618	0.628	0.629	0.627	0.626	0.625	0.622	0.628	0.0000
	Rt	3.464	3.482	3.497	3.238	3.206	3.230	3.522	3.540	3.417	3.251	3.324	3.331	3.485	3.590	3.618	3.413	0.0192
	Rsm	197.833	194.917	195.917	195.500	196.417	195.917	194.500	193.250	194.583	198.833	197.917	198.917	193.333	191.833	192.250	195.461	5.0842
9	Ra	0.151	0.153	0.148	0.138	0.140	0.140	0.147	0.148	0.146	0.134	0.135	0.134				0.143	0.0000
	Ry	1.057	1.032	1.019	1.052	1.037	1.048	1.090	1.111	1.092	0.993	0.974	0.980	1.108	1.088	1.075	1.050	0.0020
	Rz	0.711	0.706	0.697	0.760	0.759	0.752	0.755	0.767	0.758	0.734	0.735	0.733	0.823	0.813	0.802	0.754	0.0014
	Rq	0.191	0.188	0.185	0.178	0.175	0.178	0.184	0.187	0.183	0.171	0.167	0.167	0.197	0.194	0.192	0.182	0.0001
	Rt	1.442	1.444	1.428	1.461	1.431	1.535	1.527	1.547	1.556	1.306	1.274	1.299	1.490	1.432	1.406	1.438	0.0079
	Rsm	110.500	111.167	106.250	79.000	75.500	77.167	87.000	95.333	89.500	82.750	81.000	78.167	71.750	68.750	70.167	85.600	201.9397
10	Ra	0.402	0.403	0.402	0.436	0.433	0.435	0.471	0.469	0.472	0.451	0.451	0.453				0.440	0.0007
	Ry	2.411	2.408	2.395	2.227	2.193	2.183	2.473	2.483	2.468	2.485	2.486	2.467	2.419	2.412	2.364	2.392	0.0112
	Rz	1.606	1.603	1.588	1.662	1.644	1.647	1.892	1.892	1.919	1.893	1.871	1.875	1.793	1.791	1.776	1.763	0.0157
	Rq	0.488	0.491	0.487	0.512	0.507	0.509	0.560	0.558	0.558	0.543	0.544	0.543	0.533	0.533	0.529	0.526	0.0007
	Rt	3.547	3.541	3.553	2.766	2.730	2.738	3.062	3.032	2.985	3.250	3.262	3.213	2.953	2.914	2.796	3.089	0.0862
	Rsm	190.000	189.250	187.667	196.083	194.750	193.000	194.417	190.333	190.000	164.833	161.000	166.000	111.833	107.750	108.917	169.722	1101.8806
11	Ra	0.195	0.199	0.197	0.189	0.188	0.188	0.181	0.186	0.183	0.187	0.187	0.185				0.189	0.0000
	Ry	1.510	1.488	1.473	1.301	1.308	1.293	1.233	1.317	1.295	1.277	1.273	1.282	1.465	1.450	1.444	1.361	0.0094
	Rz	1.009	1.002	1.004	0.983	0.987	0.988	0.916	0.947	0.934	0.943	0.942	0.934	1.151	1.152	1.149	1.003	0.0067
	Rq	0.249	0.252	0.253	0.233	0.234	0.233	0.223	0.233	0.232	0.232	0.231	0.232	0.235	0.234	0.233	0.236	0.0001
	Rt	2.211	2.177	2.145	1.744	1.702	1.656	1.557	1.805	1.766	1.612	1.661	1.678	1.812	1.791	1.732	1.803	0.0427
	Rsm	68.083	65.917	61.583	49.917	50.083	48.667	64.000	69.500	66.583	65.417	68.500	69.250	46.250	45.167	44.667	58.906	99.5921

Tabela A.14 – Conjunto de Treinamento 1.

Rug.	2	4	6	8	10	12	14	Run
Ra	0.153	0.138	0.140	0.148	0.134	0.134	*	1
Ry	1.032	1.052	1.048	1.111	0.993	0.980	1.088	1
Rz	0.706	0.760	0.752	0.767	0.734	0.733	0.813	1
Rq	0.188	0.178	0.178	0.187	0.171	0.167	0.194	1
Rt	1.444	1.461	1.535	1.547	1.306	1.299	1.432	1
Rsm	111.167	79.000	77.167	95.333	82.750	78.167	68.750	1
Ra	0.250	0.243	0.246	0.266	0.254	0.251	*	2
Ry	1.862	1.821	1.933	1.773	1.775	1.825	2.118	2
Rz	1.245	1.213	1.247	1.240	1.288	1.296	1.414	2
Rq	0.317	0.310	0.318	0.327	0.316	0.316	0.357	2
Rt	2.576	2.688	2.908	2.403	2.467	2.626	2.953	2
Rsm	67.833	73.750	77.083	78.083	71.000	70.667	72.083	2
Ra	0.403	0.436	0.435	0.469	0.451	0.453	*	3
Ry	2.408	2.227	2.183	2.483	2.485	2.467	2.412	3
Rz	1.603	1.662	1.647	1.892	1.893	1.875	1.791	3
Rq	0.491	0.512	0.509	0.558	0.543	0.543	0.533	3
Rt	3.541	2.766	2.738	3.032	3.250	3.213	2.914	3
Rsm	189.250	196.083	193.000	190.333	164.833	166.000	107.750	3
Ra	0.480	0.477	0.474	0.453	0.348	0.348	*	4
Ry	2.425	2.484	2.523	2.455	1.941	1.957	*	4
Rz	1.879	1.850	1.853	1.783	1.464	1.468	*	4
Rq	0.572	0.572	0.573	0.554	0.429	0.428	*	4
Rt	2.901	3.258	3.323	3.158	2.451	2.487	*	4
Rsm	141.417	123.083	122.750	104.833	105.750	104.250	*	4
Ra	0.199	0.189	0.188	0.186	0.187	0.185	*	5
Ry	1.488	1.301	1.293	1.317	1.277	1.282	1.450	5
Rz	1.002	0.983	0.988	0.947	0.943	0.934	1.152	5
Rq	0.252	0.233	0.233	0.233	0.232	0.232	0.234	5
Rt	2.177	1.744	1.656	1.805	1.612	1.678	1.791	5
Rsm	65.917	49.917	48.667	69.500	65.417	69.250	45.167	5

Rug.	2	4	6	8	10	12	14	Run
Ra	0.195	0.180	0.178	0.192	0.180	0.173	*	6
Ry	1.431	1.293	1.279	1.507	1.503	1.457	1.399	6
Rz	0.932	0.919	0.897	1.028	1.041	1.021	0.921	6
Rq	0.243	0.227	0.225	0.249	0.241	0.230	0.258	6
Rt	2.003	2.103	2.127	2.175	2.374	2.296	1.889	6
Rsm	111.417	79.583	72.833	53.083	69.250	62.500	136.083	6
Ra	0.528	0.525	0.526	0.516	0.520	0.521	*	8
Ry	2.978	2.838	2.840	2.940	2.813	2.837	2.935	8
Rz	2.270	2.238	2.249	2.208	2.220	2.228	2.240	8
Rq	0.638	0.629	0.630	0.623	0.628	0.627	0.625	8
Rt	3.482	3.238	3.230	3.540	3.251	3.331	3.590	8
Rsm	194.917	195.500	195.917	193.250	198.833	198.917	191.833	8
Ra	0.326	0.308	0.275	0.253	0.240	0.258	*	9
Ry	2.376	2.148	1.914	1.586	1.772	1.898	1.431	9
Rz	1.531	1.414	1.283	1.065	1.269	1.219	1.093	9
Rq	0.425	0.397	0.341	0.308	0.319	0.328	0.286	9
Rt	3.635	3.080	2.363	2.125	3.493	2.876	1.993	9
Rsm	141.333	105.417	80.917	139.917	119.583	85.917	72.333	9
Ra	0.513	0.504	0.504	0.512	0.498	0.494	*	12
Ry	2.728	2.761	2.735	2.554	2.605	2.582	2.488	12
Rz	2.121	2.222	2.218	2.010	2.052	2.018	1.900	12
Rq	0.615	0.613	0.613	0.598	0.586	0.583	0.575	12
Rt	3.348	3.242	3.269	2.918	3.041	2.995	2.947	12
Rsm	220.750	224.333	220.667	209.917	208.083	212.333	204.000	12
Ra	0.162	0.172	0.171	0.142	0.178	0.173	*	14
Ry	1.076	1.328	1.331	1.023	1.268	1.207	1.166	14
Rz	0.720	0.852	0.858	0.743	0.893	0.875	0.862	14
Rq	0.203	0.220	0.220	0.178	0.220	0.214	0.198	14
Rt	1.513	1.877	1.879	1.286	1.612	1.525	1.522	14
Rsm	116.333	93.417	87.667	78.583	85.917	88.500	74.417	14

ANEXO B - ALGORITMO PARA O CÁLCULO DO PERFIL DE ENERGIA

$Passo = E_T/K + 1$

$Limiar = 0.0$

$Parcial = 2 \times S(0)$

$ParcialAux = 0.0$

$j = 1$

Para i de 0 até $K - 1$

$Limiar = Limiar + Passo$

Enquanto $Parcial \leq Limiar$

$ParcialAux = Parcial$

$Parcial = Parcial + S(j)$

$j = j + 1$

Fim Enquanto

Se $Parcial > Limiar$

$Aux1 = (j - 2) \times (F_s/N)$

$Aux2 = (j - 1) \times (F_s/N)$

$Perfil(i) = InterpolaçãoLinear(Aux1, ParcialAux, Aux2, Parcial, Limiar)$

Senão

$Perfil(i) = (j - 1) \times (F_s/N)$

Fim Se

Fim Para

ANEXO C - SOFTWARE PARA EXTRAÇÃO DE PARÂMETROS

```
/*
 * fft.h
 *
 *
 *      Author: edielson
 */

#ifndef FFT_H_
#define FFT_H_

void fft(double *temp_DATA, double *fft_DATA, int M, int fft_SIZE, int
window_SIZE);
#endif /* FFT_H_ */


/*
Função: fft.c
Calcula a FFT de N pontos de um dados de entrada e retorna o quadrado do
módulo da FFT

Edielson Prevato Frigieri
-------------------------------------------------------------
Parâmetros de entrada:
float *temp_DATA => vetor para calculo da FFT
float *fft_DATA => resultado da FFT
DWORD fft_SIZE => tamanho da FFT
DWORD window_SIZE => tamanho do quadro de 20ms

Parâmetros de Saída:
void
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "types.h"


void fft(double *temp_DATA, double *fft_DATA, int M, int fft_SIZE, int
window_SIZE)
{
       double Tr, Ti, Ur, Ui, Wr, Wi, b;    // var auxiliares para o calculo
da FFT
       int ip, n1, n2, n3;                  // var aux para o calculo da FFT
       register int i, j, k;                // var auxiliares para loops
       double *Xr,*Xi;                     // partes real e imaginaria da FFT

       /*-------------------------------------------------------------*/
       //Aloca memória para dados float
       Xr = (double*) malloc (sizeof(double) * (fft_SIZE));
       if(!Xr)
       {
            printf("\nError");
            return;
       }
       /*-------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*/
//Aloca memória para dados float
Xi = (double *) malloc (sizeof(double) * (fft_SIZE));

if(!Xi)
{
     printf("\nError");
     return;
}
/*-------------------------------------------------------------*/

Tr = Ti = Ur = Ui = Wr = Wi = b = 0.0;
ip = n1 = n2 = n3 = 0;

/* Calculo da FFT de N pontos do vetor temp_DATA. Usa-se o
     algoritmo proposto no Prob 6.4, pagina 333, do livro
     "Digital Signal Processing" de Oppenheim & Schaffer.
     Nomenclatura usada no livro e aqui: LE= n1, LE1= n2,
     NV2=n3.
*/

/*-------------------------------------------------------------*/
for(k=0; k<fft_SIZE; k++) // inicializacao
{
     Xr[k] = 0.0;
     Xi[k] = 0.0;
}
for(k=0; k<window_SIZE; k++)
     Xr[k] = temp_DATA[k];

// Embaralhamento da sequencia de entrada
n3 = fft_SIZE / 2;
j = 0;
for(i=0; i<fft_SIZE-1; i++) // comeco do "for"
{
     if(i < j)
     {
           Tr = Xr[j];
           Xr[j] = Xr[i];
           Xr[i] = Tr;
           /* Ti = Xi[j];      // para seq.complexas
                 Xi[j] = Xi[i];
           Xi[i] = Ti;  */
     }
     k = n3;
     while(k <= j)
     {
           j -= k;
           k /= 2;
     }
     j += k;
}     // fim do "for" (e do embaralhamento)
/*-------------------------------------------------------------*/

//-------------------------------
// Calculo da FFT
//-------------------------------
/*-------------------------------------------------------------*/
for(k=1; k<=M; k++) // "for 1"
```

```
        {
             n1 = 1;
             for(i=1; i<=k; i++) // equivale a fazer 2**k
                   n1 *= 2;
             n2 = n1 / 2;
             Ur = 1.0;
             Ui = 0.0;
             b = M_PI / n2;    // M_PI=3.1415... (definido em <math.h>)
             Wr = cos(b);
             Wi = -sin(b);

             for(j=0; j<n2; j++) // "for 2"
             {
                   for(i=j; i<fft_SIZE; i+=n1) // "for 3"
                   {
                         ip = i + n2;
                         Tr = Xr[ip] * Ur - Xi[ip] * Ui;
                         Ti = Xr[ip] * Ui + Xi[ip] * Ur;
                         Xr[ip] = Xr[i] - Tr;
                         Xi[ip] = Xi[i] - Ti;
                         Xr[i] =  Xr[i] + Tr;
                         Xi[i]  = Xi[i] + Ti;
                   }     // fim "for 3"
                   Tr = Ur;
                   Ur = Ur * Wr - Ui * Wi;
             Ui = Tr * Wi + Ui * Wr;
             }        // fim "for 2"
        }           // fim "for 1"
        // Fim do calculo da FFT. A seq. obtida esta na ordem sequencial
        /*-------------------------------------------------------------*/

        // Calculo do quadrado do modulo da FFT (equivale a energia)
        /*-------------------------------------------------------------*/
        for(k=0; k<fft_SIZE/2; k++)
             fft_DATA[k] = Xr[k] * Xr[k] + Xi[k] * Xi[k];
        /*-------------------------------------------------------------*/

        // Desalocando ponteiros
        free(Xr);
        free(Xi);
}       // fim da funcao "fft()"


/*
 * pre_enfase.h
 *
 *  Created on: 14/11/2011
 *      Author: edielson
 */

#ifndef PRE_ENFASE_H_
#define PRE_ENFASE_H_

void pre_enfase(double *fDATA, double *eDATA, long eSIZE);

#endif /* PRE_ENFASE_H_ */
```

```
/*
Função: pre_enfase.c
Efetuar o pré-ênfase no sinal wave

Edielson Prevato Frigieri
---------------------------------------------------------------
Parâmetros de entrada:
float *fDATA => dados wave para pré-ênfase
float *eDATA => dados após pré-ênfase
DWORD eSIZE => tamanho do vetor wave

Parâmetros de Saída:
void
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "types.h"

void pre_enfase(double *fDATA, double *eDATA, long eSIZE)
{
       int i; //contador

       eDATA[0]=fDATA[0];
       for (i=1;i<eSIZE;i++)
       {
            eDATA[i]=fDATA[i]-0.95*fDATA[i-1];
       }
}


/*
 * hamming.h
 *
 *
 *      Author: edielson
 */

#ifndef HAMMING_H_
#define HAMMING_H_

void hamming(double* pdHamming, long N);

#endif /* HAMMING_H_ */


/*
Função: hamming.c
Gera uma janela de hamming com o tamanho desejado

Edielson Prevato Frigieri
---------------------------------------------------------------
Parâmetros de entrada:
float *wHAMMING => vetor para guardar os valores da janela de Hamming;
DWORD N => número de amostras em 20ms;

Parâmetros de Saída:
```

```
void
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "types.h"

void hamming(double* pdHamming, long N)
{
	long i;

	for(i=0;i<N;i++)
	{
		pdHamming[i] = 0.54 - 0.46*cos((2*M_PI*i)/(N-1));
	}
}


/*
 * InterpLinear.h
 *
 *
 *	 Author: edielson
 */

#ifndef INTERPLINEAR_H_
#define INTERPLINEAR_H_

double InterpLinear(double x1,double y1, double x2, double y2, double y);

#endif /* INTERPLINEAR_H_ */


/*
 * InterpLinear.c
 *
 *
 *	 Author: edielson
 */

double InterpLinear(double x1,double y1, double x2, double y2, double y)
{
	// x1 componente x do ponto 1
	// y1 componente y do ponto 1
	// x2 componente x do ponto 2
	// y2 componente y do ponto 2
	// y  componente y do ponto intermediario


	double a; // coeficiente angular da reta que passa pelos pontos 1 e 2
	double b; // termo independente da reta que passa pelos pontos 1 e 2
	double aux_x, aux_y;   // variaveis auxiliares
	double x;              // valor de retorno da funcao

	aux_x = x1 - x2;
	aux_y = y1 - y2;

	a = aux_y / aux_x;
```

```
	b = y1-a*x1;

	x = (y - b)/a;

	return (x);
}


/*
 * MelParameters.h
 *
 *
 *      Author: edielsonpf
 */

#ifndef MELPARAMETERS_H_
#define MELPARAMETERS_H_

void MelParameters_melFilters(double* mel_DATA, double* fft_DATA, int
fft_SIZE, int filtros_SIZE, int iSampleFreq);
void MelParameters_melCepstrals(double* mel_DATA, double* mel_ceps_DATA, int
P, int filtros_SIZE);

#endif /* MELPARAMETERS_H_ */


/*
 * MelParameters.c
 *
 *
 *      Author: edielsonpf
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

double MelParamenters_roundx(double value);

/*
Função: filtros_mel.c
Passa o sinal após a FFT no banco de filtros Mel

Edielson Prevato Frigieri
-------------------------------------------------------------
Parâmetros de entrada:
double *mel_DATA => dados após os filtros mel que serão retornados para a
função chamadora
double *fft_DATA => dados após a FFT
int fft_SIZE => tamanho da FFT
int filtros_SIZE => tamanho do banco de filtros mel

Parâmetros de Saída:
void
*/

void MelParameters_melFilters(double *mel_DATA, double *fft_DATA, int
fft_SIZE, int filtros_SIZE, int iSampleFreq)
{
	double* pdCenterFreq;		//Frequencias Centrais dos Filtros Mel
```

```
        double f1,f2,f3;       //frequências para cálculo da equação da reta
        double aC, bC;         //coeficientes da equação da reta crescente
        double aD, bD;         //coeficientes da equação da reta decrescente
        double Hz_DATA;        //converte dado para Hz
        double dfft_SIZE;      //tamanho da fft em formato double
        int i,j;         //contadores


        /*-------------------------------------------------------------------*/
        //Aloca memória para dados double
        pdCenterFreq = (double *) malloc (sizeof(double) * (filtros_SIZE+1));
        if(!pdCenterFreq)
        {
             printf("\nError allocating memory");
             return;
        }
        /*-------------------------------------------------------------------*/

        //Converte fft_SIZE para double
        /*-------------------------------------------------------------------*/
        dfft_SIZE=fft_SIZE;
        /*-------------------------------------------------------------------*/

        //Inicializa frequências centrais
        /*-------------------------------------------------------------------*/
        pdCenterFreq[0]=100;
        for(i=1;i<filtros_SIZE+1;i++)
        {
             if(i<10)
                   pdCenterFreq[i]=pdCenterFreq[i-1]+100;
             else
                   pdCenterFreq[i]=MelParamenters_roundx(pdCenterFreq[i-
1]*pow(2,0.2));
        }
        /*-------------------------------------------------------------------*/

        //Calcula os filtros
        /*-------------------------------------------------------------------*/
        for (i=0;i<filtros_SIZE;i++)
        {
             //============================
             //Coeficientes Equaçao da Reta
             if(i==0)
             {
                   f1=0;
                   f2=pdCenterFreq[i];
                   f3=pdCenterFreq[i+1];
             }
        /*   else if(i==(filtros_SIZE-1))
             {
                   f1=pdCenterFreq[i-1];
                   f2=pdCenterFreq[i];
                   f3=5278;
             }     */
             else
             {
                   f1=pdCenterFreq[i-1];
                   f2=pdCenterFreq[i];
                   f3=pdCenterFreq[i+1];
             }
```

```
            aC=1/(f2-f1);
            bC=-f1/(f2-f1);
            aD=1/(f3-f2);
            bD=f3/(f3-f2);
            //============================

            //Zera mel_DATA
            //----------------------------
            mel_DATA[i]=0.0;
            //----------------------------

            for(j=0;j<(fft_SIZE/2);j++)
            {
//                Hz_DATA=j*(5512/((dfft_SIZE/2)-1));
                  Hz_DATA=j*((iSampleFreq/2)/((dfft_SIZE/2)-1));

                  if(f1<=Hz_DATA && Hz_DATA<=f2)

      mel_DATA[i]=mel_DATA[i]+fft_DATA[j]*(aC*Hz_DATA+bC);
                  else if(f2<Hz_DATA && Hz_DATA<=f3)
                        mel_DATA[i]=mel_DATA[i]+fft_DATA[j]*(-
aD*Hz_DATA+bD);
            }
      }
      /*-------------------------------------------------------------*/
      free(pdCenterFreq);
}


void MelParameters_melCepstrals(double* mel_DATA, double* mel_ceps_DATA, int
P, int filtros_SIZE)
{
      int i,m;   //contador
      double aux;      //variável auxiliar para calculo do mel ceps

      //Zera mel_ceps_DATA
      /*-------------------------------------------------------------*/
      for(i=0;i<P;i++)
      {
            mel_ceps_DATA[i]=0.0;
      }
      /*-------------------------------------------------------------*/

      //Calculo do Mel Cepstrais
      /*-------------------------------------------------------------*/
      for(m=0;m<P;m++)
      {
            aux=0.0;

            for(i=0;i<filtros_SIZE;i++)
            {

      aux=aux+log(mel_DATA[i])*cos(((M_PI*m)/(2*filtros_SIZE))*(2*i+1));
//
      printf("\r\nmel_data[%d]=%f,aux[%d]=%f",i,mel_DATA[i],i,aux);
            }
            mel_ceps_DATA[m]=aux;
      }
}
```

```
double MelParamenters_roundx(double value)
{
    if (value < 0)
        return -(floor(-value + 0.5));
    else
        return   floor( value + 0.5);
}


/*
 * perfil.h
 *
 *
 *      Author: edielsonpf
 */

#ifndef PERFIL_H_
#define PERFIL_H_

void PerfilCalc_extract(double* pdEnergy, int iVecSize , double* pdPerfil,
int iNumPerfil, int iSampleFreq);
void PerfilCalc_mean(double** pdPerfil, FILE* pOutputFile, int iNumPerfil,
int iNumQuadros, int iSampleFreq);
#endif /* PERFIL_H_ */


/*
 * perfil.c
 *
 *
 *      Author: edielsonpf
 */
#include <stdio.h>
#include <math.h>
#include "InterpLinear.h"

#define PERFILCALC_MEAN_INTERVAL    30
#define PERFILCALC_MEAN_STEP        5

void PerfilCalc_extract(double* pdModFFT, int iNumPoints , double* pdPerfil,
int iNumPerfil, int iSampleFreq)
{
       if(!pdModFFT)    return;
       if(!pdPerfil)    return;

       int  iCounterI;
       int  iCounterJ;
       double     dTotalEnergy = 0;
       double     dStep;
       double     dLimiar;
       double     dPartialB;
       double     dPartial;
       double     dAux1;
       double     dAux2;

//     dTotalEnergy = pdModFFT[0]+pdModFFT[iNumPoints/2];
//     for(iCounterI=1; iCounterI < iNumPoints/2; iCounterI++)
       for(iCounterI=0; iCounterI < iNumPoints/2; iCounterI++)
       {
```

```
            dTotalEnergy += 2*pdModFFT[iCounterI];
        }

        dStep = dTotalEnergy / (double)(iNumPerfil + 1);
        dLimiar = 0.0;
        dPartialB = 0.0;
        dPartial = 2*pdModFFT[0];

        iCounterJ = 1;
        for (iCounterI = 0; iCounterI < iNumPerfil; iCounterI++)
        {
            dLimiar += dStep;
            while (dPartial <= dLimiar)
            {
                dPartialB = dPartial;
                dPartial += 2*pdModFFT[iCounterJ];
                iCounterJ++;
            }
            if(dPartial > dLimiar) // passou do ponto
            {

                dAux1=((double)iCounterJ-
        2.0)*(double)iSampleFreq/(double)iNumPoints;  // em Hertz
                dAux2=((double)iCounterJ-
        1.0)*(double)iSampleFreq/(double)iNumPoints;  // em Hertz
                pdPerfil[iCounterI]=
        InterpLinear(dAux1,dPartialB,dAux2,dPartial,dLimiar);
            }
            else
            {
                pdPerfil[iCounterI]=((double)iCounterJ-
        1.0)*(double)iSampleFreq/(double)iNumPoints;; // em Hertz
            }
        }
}

void PerfilCalc_mean(double** pdPerfil, FILE* pOutputFile, int iNumPerfil,
int iNumQuadros, int iSampleFreq)
{
        if(!pdPerfil)          return;

        int  iInicio;
        int  iFim;
        int  iCounter;
        int  iAux1,iAux2;
        double     dCalcAux;

        iInicio=0;
        iFim=PERFILCALC_MEAN_INTERVAL;
        iCounter=0;

        while(iFim < iNumQuadros)
        {
            for(iAux1=0;iAux1<iNumPerfil;iAux1++)
            {
                dCalcAux=0;

                for(iAux2=iInicio; iAux2<iFim;iAux2++)
                {
                    dCalcAux+=pdPerfil[iAux2][iAux1];
```

```
                    }
//                  dCalcAux/=(PERFILCALC_MEAN_INTERVAL*2*M_PI);
                    dCalcAux/=(PERFILCALC_MEAN_INTERVAL*(iSampleFreq/2));
//                  dCalcAux/=(PERFILCALC_MEAN_INTERVAL);
                    fwrite(&dCalcAux,sizeof(double),1,pOutputFile);
                }
                iInicio+= PERFILCALC_MEAN_STEP;
                iFim+= PERFILCALC_MEAN_STEP;
                iCounter++;
        }
}
```

ANEXO D - SOFTWARE PARA TREINAMENTO DE MODELOS GMM

```
/*
 * mixture.h
 *
 */
#ifndef MIXTURE_H_
#define MIXTURE_H_

// Calculate the a posteriori probabilities
double mixture
(
  double *x,          // feature vector
  int i,              // id of Gaussian under analysis
  struct mixture *l, // GMM models
  int nGaussians,     // number of Gaussians in the mixture
  int dim             // dimension of the acoustic vectors
);

#endif /*MIXTURE_H_*/


/*
 * mixture.c
 *
 *      Calculate the a posteriori probabilities
 */

#include "structs.h"
#include "b.h"

double mixture
(
  double *x,          // feature vector
  int i,              // id of Gaussian under analysis
  struct mixture *l, // GMM models
  int nGaussians,     // number of Gaussians in the mixture
  int dim             // dimension of the acoustic vectors
)
{
        double num, den; // aux variables
        int j;           // counter

        num = l[i].p*b(x,l[i].m,l[i].s,dim);

    den = 0.0;
    for (j=0;j<nGaussians;j++)
    {
       den += l[j].p * b(x,l[j].m,l[j].s,dim);
    }
    return num/den;
}


/*
 * update_s.h
 *
 */
```

```
#ifndef UPDATE_S_H_
#define UPDATE_S_H_

void update_s
(
       double **x,         // sequence of feature vectors x[t][i]
       int dim,            // dimension of feature vectors
       int nFrames,        // number of feature vectors in x
       struct mixture *l, // GMMs
       int nGaussians,     // number of Gaussians in the mixture
       double **m,         // gaussian means m[nGaussians][dim]
       double ***s1        // gaussian variances s[nGaussians][dim]
);

#endif /*UPDATE_S_H_*/


/*
 * update_s.c
 *
 * Update the mixture coefficients
 */

#include <stdlib.h>

#include "structs.h"
#include "mixture.h"

void update_s
(
       double **x,         // sequence of feature vectors x[t][i]
       int dim,            // dimension of feature vectors
       int nFrames,        // number of feature vectors in x
       struct mixture *l, // GMMs
       int nGaussians,     // number of Gaussians in the mixture
       double **m,         // gaussian means m[nGaussians][dim]
       double ***s1        // gaussian variances s[nGaussians][dim]
)
{
       double **s;        // Gaussian variances
       double *num, den, aux; // aux variables
    int i,j,t;           // counters
    double threshold = 0.001; // minimum values allowed for the variances

    s = *s1;

       // Allocating memory
       num = malloc(sizeof(double)*dim);

       // Updating Gaussian means
    for (i=0;i<nGaussians;i++)
    {
       den = 0.0;
       for (j=0;j<dim;j++)
            num[j] = 0.0;

       for (t=0;t<nFrames;t++)
       {
            aux = mixture(x[t],i,l,nGaussians,dim);
            den += aux;
```

```
            for (j=0;j<dim;j++)
                        num[j] += aux*x[t][j]*x[t][j];

        }
        for (j=0;j<dim;j++)
            s[i][j] = num[j]/den-m[i][j]*m[i][j];
    }

    // Limiting the values of the variances
    for(i=0;i<nGaussians;i++)
        for(j=0;j<dim;j++)
            if(s[i][j] < threshold)
                  s[i][j] = threshold;

    // Freeing memory
    free (num);

    // Returning updated variances
    *s1 = s;
}


/*
 *      update_p.h
 *
 *
 */

#ifndef UPDATE_P_H_
#define UPDATE_P_H_

void update_p
(
        double **x, // sequence of feature vectors x[t][i]
        int dim,    // dimension of feature vectors
        int nFrames,    // number of feature vectors in x
        struct mixture *l,          // GMMs
        int nGaussians,  // number of Gaussians in the mixture
        double *p
);

#endif /*UPDATE_P_H_*/


/*
 *      update_p.c
 *
 *      Update the mixture coefficients
 *
 */

#include <stdlib.h>

#include "structs.h"
#include "mixture.h"

void update_p
(
        double **x, // sequence of feature vectors x[t][i]
```

```
        int dim,    // dimension of feature vectors
        int nFrames,    // number of feature vectors in x
        struct mixture *l,          // GMMs
        int nGaussians,  // number of Gaussians in the mixture
        double *p
)
{
    int i,t; // counters

        // Updating mixture coefficients
        for (i=0;i<nGaussians;i++)
             p[i] = 0;

        for (i=0;i<nGaussians;i++)
        {
             for (t=0;t<nFrames;t++)
                   p[i] += mixture(x[t],i,l,nGaussians,dim);

             p[i] /= (double)nFrames;
        }
}

/*
 *      update_m.h
 *
 *
 */

#ifndef UPDATE_M_H_
#define UPDATE_M_H_

void update_m
(
        double **x,         // sequence of feature vectors x[t][i]
        int dim,            // dimension of feature vectors
        int nFrames,        // number of feature vectors in x
        struct mixture *l, // GMMs
        int nGaussians,     // number of Gaussians in the mixture
        double ***m1        // gaussian means m[nGaussians][dim]
);

#endif /*UPDATE_M_H_*/


/*
 *      update_m.c
 *
 *      Update the mixture coefficients
 */

#include <stdlib.h>
#include <stdio.h>

#include "structs.h"
#include "mixture.h"

void update_m
(
        double **x,         // sequence of feature vectors x[t][i]
        int dim,            // dimension of feature vectors
```

```
        int nFrames,        // number of feature vectors in x
        struct mixture *l, // GMMs
        int nGaussians,     // number of Gaussians in the mixture
        double ***m1        // gaussian means m[nGaussians][dim]
)
{
        double **m;        // Gaussian means
        double *num, den, aux; // aux variables
    int i,j,t;           // counters

    m = *m1;

        // Allocating memory
        num = malloc(sizeof(double)*dim);

        // Updating Gaussian means
    for (i=0;i<nGaussians;i++)
    {
        //printf("gaussiana = %d\n",i);
        den = 0.0;
        for (j=0;j<dim;j++)
             num[j] = 0.0;

        for (t=0;t<nFrames;t++)
        {
             aux = mixture(x[t],i,l,nGaussians,dim);
             den += aux;

             for (j=0;j<dim;j++)
                   {
                   num[j] += aux*x[t][j];
             }
        }
        //printf("den = %f\n",den);
        for (j=0;j<dim;j++)
        {
             m[i][j] = num[j]/den;
        }
    }

    // Freeing memory
    free (num);

    // Returning updated mean
    *m1 = m;
}


/*
 *      lbg.h
 *
 */

#ifndef LBG_H_
#define LBG_H_

//-------------------------------------------------------------------------
--
// Funcao que calcula 'n_codebooks' codebooks a partir de 'n_vetores' vetores
// de treinamento. Os vetores sao de ordem 'ordem', e sao armazenados em 'x'.
```

```
// A funcao retorna os 'n_codebooks' codebooks calculados
void lbg
(
  double ***codebook1, // armazena os codebooks
  int n_codebooks, // numero de codebooks desejado para o quantizador
  int n_vetores,   // numero de vetores exemplo para o calculo dos codebooks
  int ordem,       // ordem dos vetores exemplo
  double **x       // vetores exemplo para o calculo dos codebooks
);

//-------------------------------------------------------------------------
--
// Funcao que calcula a distancia euclidiana entre dois vetores
double deucl
(
  double *p1, // vetor 1
  double *p2, // vetor 2
  int ordem        // ordem dos vetores
);

//-------------------------------------------------------------------------
--
// Funcao que retorna o centroide de um conjunto de vetores
void calcula_centroide
(
  double **centroide1, // centroide
  int n_vetores,            // numero de vetores exemplo para o calculo dos
codebooks
  int ordem,                // ordem dos vetores exemplo
  double **x                // vetores exemplo para o calculo dos codebooks
);

//-------------------------------------------------------------------------
--
//  Funcao  que  ordena  os  vetores  do  codebook  'codebook_in'  em  ordem
decrescente,
// de acordo com o numero de vetores de treinamento associado a cada um dos
// vetores codigo.
// Posteriormente, retorna em 'codebook_out' apenas os 'n_codevectors' com
maior
// numero de vetores de treinamento associados
void ordena_elimina
(
  double **codebook_in, // codebook a ser reordenado
  double ***codebook_out1, // codebook reordenado com apenas 'n_codevectors'
vetores codigo
  int n_codevectors, // numero de vetores codigo desejado
  int *numero, // numero de vetores de treinamento associados a cada vetor
codigo
  int num_vet, // numero de vetores codigo no codebook atual
  int ordem // ordem dos vetores exemplo
);

//-------------------------------------------------------------------------
--
// Realiza operacoes para o calculo de um novo codebook, depois do splitting
void novo_codebook
(
  double ***codebook1, // armazena o codebook a ser processado
  int **numero1, // armazena o numero de vetores em cada particao
```

```
  //int num_vet, // menor numero maior ou igual a n_codevectors que e tambem
potencia de 2.
  int n_vet, // tamanho atual do codebook
  int n_vetores,   // numero de vetores exemplo para o calculo dos codebooks
  int ordem,       // ordem dos vetores exemplo
  double ***somas1, // variavel auxiliar
  double **x       // vetores exemplo para o calculo dos codebooks
);

//-------------------------------------------------------------------------
--
// Funcao que associa vetores de treinamento ao vetor codigo mais proximo
void associa
(
  double **codebook, // codebook
  double *d_media1, // distancia media com este codebook
  int **numero1, // armazena o numero de vetores em cada particao
  int n_vet,         // tamanho atual do codebook
  int n_vetores,   // numero de vetores exemplo para o calculo dos codebooks
  int ordem,       // ordem dos vetores exemplo
  double ***somas1, // variavel auxiliar
  double **x       // vetores exemplo para o calculo dos codebooks
);

//-----------------------------------------------------------------
// Funcao que, dado um vetor exemplo e um codebook, verifica qual vetor
// codigo esta mais proximo do vetor exemplo.
// O valor de retorno indica o indice do vetor codigo mais proximo do
// vetor exemplo
int mais_proximo
(
  double **codebook, // codebook
  double *minimo1, // distancia euclidiana entre o vetor exemplo e o vetor
codigo mais proximo
  int n_codevectors, // numero de vetores no codebook
  int ordem, // ordem dos vetores
  double *x // vetor exemplo
);

#endif /*LBG_H_*/


/*
 *      lbg.c
 *
 */
//-------------------------------------------------------------------------
--
// Programa para gerar codebook para quantizacao vetorial
// Utiliza o algoritmo LBG em sua versao splitting
//-------------------------------------------------------------------------
--

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include "lbg.h"
```

```
//-------------------------------------------------------------------------
-----
// Funcao que calcula 'n_codevectors' codebooks a partir de 'n_vetores'
vetores
// de treinamento. Os vetores sao de ordem 'ordem', e sao armazenados em 'x'.
// A funcao retorna os 'n_codevectors' codebooks calculados
void lbg
(
  double ***codebook1, // armazena o codebook
  int n_codevectors,   // numero de codebooks desejado para o quantizador
  int n_vetores,          // numero de vetores exemplo para o calculo dos
codebooks
  int ordem,           // ordem dos vetores exemplo
  double **x           // vetores exemplo para o calculo dos codebooks
)
{
  // Declaracao de variaveis locais
  double **codebook; // armazena o codebook
  double **codebook2; // codebook gerado com 2^n vetores
  double epsilon=0.005; // perturbacao (ver demonstracao do algoritmo)
  int expoente; // var aux para calcular num_vet
  int register i,j; // contadores
  int *numero; // armazena o numero de vetores em cada particao
  int num_vet; // menor numero maior ou igual a n_codevectors que e tambem
potencia de 2.
  int n_vet; // tamanho atual do codebook
  double  *p1,*p2;  //  variaveis  auxiliares  no  calculo  de  distancias
euclidianas
  double **somas; // variavel auxiliar

//*************************************************************************
*****

  codebook = *codebook1;

  // Determinando menor numero maior ou igual a n_codevectors, e que seja
potencia
  // de 2 (pois na versao splitting, o LBG calcula apenas codebooks com numero
  // de vetores que seja potencia de 2)
  expoente = 0;
  num_vet = pow(2,expoente);
  while (num_vet < n_codevectors)
  {
    expoente++;
    num_vet = pow(2,expoente);
  }

  // Alocando memoria p/ os ponteiros

  // Codebook
  codebook2 = malloc(sizeof(double *)*num_vet);
  for (i=0;i<num_vet;i++)
      codebook2[i] = malloc(sizeof(double)*ordem);

  // Matriz auxiliar no calculo do codebook
  somas = malloc(sizeof(double *)*num_vet);
  for (i=0;i<num_vet;i++)
    somas[i] = malloc(sizeof(double)*ordem);

  // Var's auxiliares no calculo da dist. Euclidiana
```

```
  p1 = malloc(sizeof(double)*ordem);
  p2 = malloc(sizeof(double)*ordem);

  // Numero de vetores em cada particao
  numero = malloc(sizeof(int)*num_vet);

  // Calculando centroide de todos os vetores (codebook inicial)
  calcula_centroide(&codebook2[0],n_vetores,ordem,x);
  n_vet = 1;

       printf("Number of vectors: %d\n",n_vet);
  // Calculando demais codebooks
  while (n_vet < num_vet)
  {


    // Splitting matriz codebook
       for (i=0; i < n_vet; i++)
         for (j=0; j < ordem; j++)
           somas[i][j] = codebook2[i][j];

       for (i=1; i<=n_vet; i++)
         for (j=0; j < ordem; j++)
      {
           codebook2[2*(n_vet-i)][j] = somas[n_vet-i][j] - epsilon;
            codebook2[2*(n_vet-i)+1][j] = somas[n_vet-i][j] + epsilon;
         }

       // Proximo codebook
       n_vet *= 2;

       printf("Number of vectors: %d\n",n_vet);

    // Calculando novo codebook
    novo_codebook(&codebook2,&numero,n_vet,n_vetores,ordem,&somas,x);
  }

//-------------------------------------------------------------------------
-----
  // Temos agora um codebook otimizado com num_vet vetores codigo
  // Se (n_codevectors < num_vet), devemos retirar alguns vetores codigo do
codebook
  // O criterio de retirada e o numero de vetores de treinamento associados
a cada
  // um dos vetores codigo. Aqueles com menor numero de vetores de treinamento
  // associados sao eliminados primeiro.

  if (n_codevectors < num_vet)
  {
    // Eliminando vetores codigo de 'codebook2' para gerar 'codebook' com
'n_codevectors'
    // vetores codigo
    ordena_elimina(codebook2,&codebook,n_codevectors,numero,num_vet,ordem);

    // Atualizando codebook

novo_codebook(&codebook,&numero,n_codevectors,n_vetores,ordem,&somas,x);
  }
  else
    for (i=0;i<n_codevectors;i++)
```

```
      for (j=0;j<ordem;j++)
        codebook[i][j] = codebook2[i][j];

//*************************************************************************
*****

  // Desalocando ponteiros
  free(p1);
  free(p2);

  for (i=0;i<num_vet;i++)
  {
    free(somas[i]);
    free(codebook2[i]);
  }
  free(somas);
  free(codebook2);

  free(numero);

  // Retornando codebook calculado
  *codebook1 = codebook;
}

//*************************************************************************
*****
// Funcao que calcula a distancia euclidiana entre dois vetores
double deucl
(
  double *p1, // vetor 1
  double *p2, // vetor 2
  int ordem   // ordem dos vetores
)
{
  int i;
  double soma;

  soma = 0.0;
  for (i=0; i<ordem; i++)
       soma += (p1[i]-p2[i])*(p1[i]-p2[i]);

  return sqrt(soma);
}

//*************************************************************************
*****
// Funcao que retorna o centroide de um conjunto de vetores
void calcula_centroide
(
  double **centroide1, // centroide
  int n_vetores,             // numero de vetores exemplo para o calculo dos
codebooks
  int ordem,                 // ordem dos vetores exemplo
  double **x                 // vetores exemplo para o calculo dos codebooks
)
{
  // Declaracao das variaveis locais
  double *centroide; // centroide dos vetores exemplo
  int i,j; // contadores
```

```
  centroide = *centroide1;

  // Inicializando variavel que ira conter o centroide
  for (i=0;i<ordem;i++)
       centroide[i]=0.0;

  // Somando os quadros
  for (i=0;i<n_vetores;i++)
    for (j=0;j<ordem;j++)
      centroide[j] += x[i][j];

  for (i=0;i<ordem;i++)
    centroide[i] /= (float)n_vetores;

  *centroide1 = centroide;

}
//*********************************************************************
*****
//  Funcao que ordena os vetores do codebook 'codebook_in' em ordem
decrescente,
// de acordo com o numero de vetores de treinamento associado a cada um dos
// vetores codigo.
// Posteriormente, retorna em 'codebook_out' apenas os 'n_codevectors' com
maior
// numero de vetores de treinamento associados
void ordena_elimina
(
  double **codebook_in, // codebook a ser reordenado
  double ***codebook_out1, // codebook reordenado com apenas 'n_codevectors'
vetores codigo
  int n_codevectors, // numero de vetores codigo desejado
  int *numero, // numero de vetores de treinamento associados a cada vetor
codigo
  int num_vet, // numero de vetores codigo no codebook atual
  int ordem // ordem dos vetores exemplo
)
{
  double **codebook_out; // codebook reordenado com apenas 'n_codevectors'
vetores codigo
  int done = 0; // flag auxilliar para fim de processamento
  register int i,j,k; // contadores
  int  *index;  // indices dos vetores codigo do codebook (var aux para
ordenacao)

  codebook_out = *codebook_out1;

  // Alocando memoria
  index = malloc(sizeof(int)*num_vet);

  for (i=0;i<num_vet;i++)
    index[i] = i;

  while (!done)
  {
    done = 1;
    for (i=0;i<(num_vet-1);i++)
    {
      j = index[i];
      k = index[i+1];
```

```
      if (numero[j] < numero[k])
      {
        index[i] = k;
        index[i+1] = j;
        done = 0;
      }
    }
  }

  // Salvando 'n_codevectors' em 'codebook_out'
  for (i=0;i<n_codevectors;i++)
    for (j=0;j<ordem;j++)
      codebook_out[i][j] = codebook_in[index[i]][j];

  // Desalocando ponteiro
  free(index);

  // Codebook reduzido
  *codebook_out1 = codebook_out;
}

//---------------------------------------------------------------------------
-----
// Realiza operacoes para o calculo de um novo codebook, depois do splitting
void novo_codebook
(
  double ***codebook1, // armazena o codebook a ser processado
  int **numero1, // armazena o numero de vetores em cada particao
  int n_vet, // tamanho atual do codebook
  int n_vetores,   // numero de vetores exemplo para o calculo dos codebooks
  int ordem,       // ordem dos vetores exemplo
  double ***somas1, // variavel auxiliar
  double **x       // vetores exemplo para o calculo dos codebooks
)
{
  double **codebook; // // armazena o codebook a ser processado
  double distorcao; // verifica se o algoritmo convergiu
  double d_media[2]; // distancia media ao realizar a quantizacao com o
codebook
  int register i,j,n; // contadores
  int *numero; // armazena o numero de vetores em cada particao
  double **somas; // variavel auxiliar

  numero = *numero1;
  somas = *somas1;
  codebook = *codebook1;

  // Inicializando vetor que armazena a distancia media
  d_media[0] = d_media[1] = 1e20;

  // Inicializando a variavel numero (conta quantos vetores em cada particao)
  for (i=0;i<n_vet;i++)
    numero[i] = 0;

  // Inicializando a variavel somas (var aux p/ calcular o centroide de cada
particao)
  for (i=0;i<ordem;i++)
    for (j=0;j<n_vet;j++)
      somas[j][i] = 0.0;
```

```
  // Associando vetores a particoes
  associa(codebook,&d_media[1],&numero,n_vet,n_vetores,ordem,&somas,x);

  distorcao = 1;
  n=0;
  while (distorcao > 0.005)
  {
    // Calculando o centroide de cada particao
    for (i=0;i<n_vet;i++)
      for (j=0;j<ordem;j++)
        codebook[i][j] = somas[i][j]/numero[i];

    // Inicializando a variavel numero (conta quantos vetores em cada
particao)
    for (i=0;i<n_vet;i++)
      numero[i] = 0;

    // Inicializando a variavel somas (var aux p/ calcular o centroide de
cada particao)
    for (i=0;i<ordem;i++)
      for (j=0;j<n_vet;j++)
        somas[j][i] = 0.0;

    // Calculando a distancia media com este codebook
    d_media[0] = d_media[1];
    d_media[1] = 0.0;

    // Associando vetores a particoes
    associa(codebook,&d_media[1],&numero,n_vet,n_vetores,ordem,&somas,x);

    d_media[1] /= (float)n_vetores;

    // Calculando a distorcao
    if (d_media[1] != 0)
      distorcao = (d_media[0]-d_media[1])/d_media[1];
    else
      distorcao = 0; // se a distancia media e zero, temos um quantizador
otimo
    if (distorcao < 0.0)
      distorcao=1;
    n++;
  }

  // Valores de retorno da funcao
  *codebook1 = codebook;
  *numero1 = numero;
  *somas1 = somas;
}

//-------------------------------------------------------------------------
-----
// Funcao que associa vetores de treinamento ao vetor codigo mais proximo
void associa
(
  double **codebook, // codebook
  double *d_media1, // distancia media com este codebook
  int **numero1, // armazena o numero de vetores em cada particao
  int n_vet,          // tamanho atual do codebook
  int n_vetores,   // numero de vetores exemplo para o calculo dos codebooks
  int ordem,       // ordem dos vetores exemplo
```

```
  double ***somas1, // variavel auxiliar
  double **x       // vetores exemplo para o calculo dos codebooks
)
{
  //double distancia; // distancia euclidiana entre dois vetores
  double d_media; // distancia media com este codebook
  int register i,l; // contadores
  double minimo; // menor distancia
  int *numero; // armazena o numero de vetores em cada particao
  int qual; // vetor codigo associado ao vetor de treinamento atual
  double **somas; // variavel auxiliar

  somas = *somas1;
  numero = *numero1;
  d_media = *d_media1;

  // Calculando distancias dos parametros aos centroides
  for (i=0;i<n_vetores;i++)
  {
    // Verificando vetor codigo mais proximo de x[i]
    qual = mais_proximo(codebook,&minimo,n_vet,ordem,x[i]);

    // Atualizando contagens dos clusters
    for (l=0;l<ordem;l++)
      somas[qual][l] += x[i][l];
    numero[qual]++;

    // Atualizando d_media
    d_media += minimo;
  }

  // Valores de retorno da funcao
  *d_media1 = d_media;
  *somas1 = somas;
  *numero1 = numero;
}

//-------------------------------------------------------------------
// Funcao que, dado um vetor 'x' e uma lista de vetores 'lista', verifica
qual vetor
// de 'lista' esta mais proximo de 'x'.
// O valor de retorno indica o indice do vetor codigo mais proximo do
// vetor exemplo
int mais_proximo
(
  double **lista, // lista de vetores com a qual o vetor 'x' ira ser analisado
  double *minimo1, // distancia euclidiana entre o vetor exemplo e o vetor
codigo mais proximo
  int n_codevectors, // numero de vetores no codebook
  int ordem, // ordem dos vetores
  double *x // vetor sob analise
)
{
  double distancia; // distancia euclidiana entre dois vetores
  int register l; //contadores
  double minimo; // distancia euclidiana entre o vetor exemplo e o vetor
codigo mais proximo
  int qual; // vetor codigo mais proximo do vetor exemplo

  minimo = *minimo1;
```

```
  minimo = 1e20;

  for (l=0;l<n_codevectors;l++)
  {
    distancia = deucl(x,lista[l],ordem);
    if (distancia < minimo)
    {
      minimo = distancia;
      qual = l;
    }
  }

  *minimo1 = minimo;
  return qual;
}
```

```
/*
 *      b.h
 *
 */

#ifndef B_H_
#define B_H_

double b
(
  double *x,       // vetor de parametros
  double *media, // vetor com as medias
  double *var,    // vetor com as variancias
  int ordem      // ordem do vetor de parametros
);

#endif /*B_H_*/
```

```
/*
 *      b.c
 *      Coomputes a dim-variate Gaussian function
 */

#include <math.h>
#include "structs.h"

// Funcao que implementa uma fdp gaussiana multidimensional
double b
(
  double *x,       // vetor de parametros
  double *media, // vetor com as medias
  double *var,    // vetor com as variancias
  int ordem      // ordem do vetor de parametros
)
{
  // Declaracao das variaveis locais
  double aux=0.0,aux1,aux2 = 0.0; // variaveis auxiliares
  double det; // determinante da matriz de covariancia
```

```
  double dif; // var aux p/ o calculo da gaussiana
  int i; // contador
  double prob=0.0; // probabilidade do vetor dada a distribuicao

  // (2*pi)^(ordem/2)
  aux1 = 2.0*PI;
  aux2 = ordem / 2.0;
  aux1 = pow(aux1,aux2);
  //aux1 = exp(aux2*log(aux1));

  // Calculando determinante da matriz de covariancia
  det = 1.0;
  for (i=0;i<ordem;i++)
    det *= var[i];

  if (det != 0)
  {
    aux2 = fabs(det);
    aux2 = pow(aux2,0.5);

    for (i=0;i<ordem;i++)
    {
      dif = x[i] - media[i];
      aux += (dif*dif)/var[i];
    }
    aux *= (-0.5);
    aux = exp(aux);

    prob = aux/(aux1*aux2);
  }

  return(prob);

}


/*
 *     aposteriori.h
 *     Calculate the a posteriori probabilities
 */

#ifndef APOSTERIORI_H_
#define APOSTERIORI_H_


double aposteriori
(
  double *x,          // feature vector
  int i,              // id of Gaussian under analysis
  struct mixture *l, // GMM models
  int nGaussians,     // number of Gaussians in the mixture
  int dim             // dimension of the acoustic vectors
);

#endif /*APOSTERIORI_H_*/


/*
 *     aposteriori.c
 *     Calculate the a posteriori probabilities
```

```
 */

#include "structs.h"
#include "b.h"

double aposteriori
(
  double *x,          // feature vector
  int i,              // id of Gaussian under analysis
  struct mixture *l, // GMM models
  int nGaussians,     // number of Gaussians in the mixture
  int dim             // dimension of the acoustic vectors
)
{
        double num, den; // aux variables
        int j;           // counter

        num = l[i].p*b(x,l[i].m,l[i].s,dim);

    den = 0.0;
    for (j=0;j<nGaussians;j++)
      den += l[j].p * b(x,l[j].m,l[j].s,dim);

    return num/den;
}
```

ANEXO E - SOFTWARE PARA TESTE DE GRUPO DE RUGOSIDADES

```
/*
 *      testgmm.h
 *
 */

#ifndef TESTGMM_H_
#define TESTGMM_H_

double testgmm
(
        double **x, // observation sequence
        int dim, //dimension of feature vectors
        int nFrames, // number of vectors in x
        struct mixture *l, // GMM
        int nGaussians
);

#endif /*TESTGMM_H_*/


/*
 *      testgmm.c
 *      Function that calculates the a posteriori probability of an
observation, given the GMM
 */

#include <math.h>

#include "structs.h"
#include "b.h"

double testgmm
(
        double **x, // observation sequence
        int dim, //dimension of feature vectors
        int nFrames, // number of vectors in x
        struct mixture *l, // GMM
        int nGaussians
)
{
        int i,t; // counters
        double p = 0.0; // a posteriori probability
        double aux;

        for (t=0;t<nFrames;t++)
        {
             aux = 0.0;
             for (i=0;i<nGaussians;i++)
                   aux += l[i].p*b(x[t],l[i].m,l[i].s,dim);
             p += log(aux);
        }
        return p;
}
```