v. 8, n. 2, p. 22-36, 2019
ISSN 2237-9223

REVISTA BRASILEIRA DE CRIMINALÍSTICA

DOI: http://dx.doi.org/10.15260/rbc.v8i2.358

Análise temporal e espectral dedicada ao reconhecimento computacional de locutores via modelamento probabilístico e determinístico

F. R. Silva [a,*], R. C. Guido [b]

[a] *Superintendência da Polícia Técnico-Científica, Instituto de Criminalística, Núcleo de Perícias Criminalísticas, São José do Rio Preto (SP), Brasil*
[b] *Instituto de Biociências, Letras e Ciências Exatas, Unesp – Univ. Estadual Paulista (São Paulo State University), São José do Rio Preto (SP), Brasil*

**Endereço de e-mail para correspondência: fabricio.frs@policiacientifica.sp.gov.br . Tel.: +55-17-991471131.*

Recebido em 28/05/2019; Revisado em 10/06/2019; Aceito em 17/06/2019

Resumo

Dentre as áreas que constituem a ciência forense, métodos computacionais estão cada vez mais ganhando destaque na análise de vestígios criminais. Um dos casos mais frequentes da perícia é a análise de telefones celulares e gravações telefônicas, onde a identificação dos agentes comunicantes é de fundamental importância para o indiciamento ou não de um suspeito. O objetivo deste trabalho é projetar e implementar um algoritmo para a identificação biométrica de indivíduos, por meio da análise de suas vozes, comparando uma abordagem probabilística com uma determinística.

Palavras-Chave: Identificação de locutores; Processamento digital de sinais; Modelo probabilístico;

Abstract

Among the areas which are forensics, computational methods are increasingly gaining importance in criminal trace analysis. One of the most frequent cases of expertise is the analysis of mobile phones and telephone recordings, where the identification of connecting agents is of fundamental importance to the indictment or not a suspect. The objective of this work is to design and implement an algorithm for biometric identification of individuals by analyzing their voices by comparing a probabilistic approach with a deterministic.

Keywords: Speaker identification; Digital signal processing; Probabilistic model;

1. INTRODUÇÃO

Devido ao constante desenvolvimento dos computadores, tanto em termos de *hardware* quanto de *software*, os sistemas computacionais passaram a despertar o interesse de diversas áreas, tanto no âmbito acadêmico quanto na área tecnológica. Nesse contexto, como aplicação tecnológica promissora, é possível destacar a identificação biométrica por voz [1]. Registre-se que diversos trabalhos acadêmicos na área de processamento digital de voz têm sido desenvolvidos [1-3] em vista do crescente interesse acerca do tema.

Nos últimos anos, as ciências forenses [4,5] passaram a englobar a computação como ciência relevante. De maneira específica, a figura ativa no emprego de tal ciência é o perito criminal. Um bom texto acerca da importância da ciência forense computacional foi escrito por Camila Lenke, redatora do portal eletrônico *under-linux.org*, e está disponível em [6].

Por outro lado, a precariedade de instalações e equipamentos, além da grande falta de profissionais na área, tem sido alvo de constante exploração da mídia[1]. Logo, atualmente, tal aspecto tem evidenciado

[1] **Perícia criminal capixaba acumula 10 anos de atraso tecnológico**. Artigo publicado no portal *Gazeta Online* em 07/05/2015, disponível em http://gazetaonline.globo.com/_conteudo/2015/05/noticias/cidades/3896388-pericia-criminal-capixaba-acumula-10-anos-de-atraso-tecnologico-afirma-associacao.html. Acessado em 24/01/2019.

negativamente o belo trabalho desenvolvido diariamente pelos órgãos periciais.

Em linhas gerais, as ciências forenses são formadas pela junção interdisciplinar de diversas áreas, tais como matemática, física, química, biologia, entre outras, com o intuito de desvendar ilícitos penais. Nesse contexto, a computação se insere como sendo a ferramenta mais comum na análise de escutas telefônicas e na perícia de telefones celulares, onde, em tais casos, identificar, ou não, a voz de um suspeito pode ser fundamental para o indiciamento, ou não, do mesmo.

Vale ressaltar que já é possível encontrar trabalhos acadêmicos abordando as áreas da ciência forense, seja computacional [7] ou físico-químico biológico [8], e, ainda enfatizando tal crescimento de interesse, o último concurso público para o cargo de perito criminal da Superintendência da Polícia Técnico-Científica, realizado no ano de 2013, apresentou grande concorrência pelas vagas, sendo 22.801 inscritos para 447 vagas[2].

Portanto, em vista das considerações expostas nos parágrafos anteriores, o objetivo do presente trabalho é o estudo de conceitos inerentes à área de processamento digital de voz para, em seguida, efetuar o projeto e a implementação de um algoritmo computacional, em linguagem de programação C/C++, dependente do diálogo e com funcionamento *off-line*, que possa auxiliar a perícia criminal por meio da identificação biométrica de indivíduos com base em seus padrões acústicos.

Finalmente, o presente texto está dividido em capítulos. O capítulo 2 trata-se de uma revisão bibliográfica acerca do embasamento teórico utilizado e o capítulo 3 descreve a metodologia, incluindo a análise dos algoritmos desenvolvidos, adotada na execução do trabalho. Os resultados, devidamente comentados, são apresentados no capítulo 4. Por fim, o capítulo 5 é reservado para as conclusões. Adicionalmente, os códigos-fonte das implementações realizadas estão no apêndice.

2. ABORDAGEM TEÓRICA

O presente capítulo, abordagem teórica, tem como finalidade auxiliar no que consiste em apresentar os conceitos utilizados no decorrer do trabalho. Assim, este capítulo em dividido em seções. A seção 2.1 apresenta uma breve explanação acerca de processamento digital de sinais. As seções 2.2 e 2.3 abordam as técnicas de amostragem e quantização. Por fim, a seção 2.4 descreve as duas técnicas utilizadas para processar sinais de áudio.

2.1. Processamento Digital de Sinais (PDS)

A análise e o estudo de mídias digitais, mais precisamente arquivos de áudio, consiste como um dos ramos de estudo da área da computação conhecida como Processamento digital de sinais.

Alavancada pelo crescente desenvolvimento computacional em termo de *hardware* e *software*, a área de processamento digital, nos dias atuais, está presente nos mais diversos equipamentos tecnológicos. Tal fato baseia-se, por exemplo, na presença indispensável de sistemas digitais de áudio em MP3 *Players*, telefones celulares, *tablets*, entre outros.

A popularização identificada no parágrafo anterior impulsionou o crescimento de estudos e pesquisas na referida área. Avanços constantes de codificadores/decodificadores de sinais digitais possibilitam, atualmente, tais processos na ordem de milissegundos, sendo que tais tempos apresentam a tendência de diminuir cada vez mais.

Adentrando a parte teórica de PDS [9], o estudo de assuntos relacionados a essa área relaciona-se fortemente com a forma como sinais de áudio são representados, uma vez que se caracterizam como sinais contínuos variando temporalmente. Assim, manipular tais tipos de sinais caracteriza-se, se analisada como sinal contínuo dependente de uma variável temporal, como uma tarefa árdua e complexa.

Portanto, a utilização de técnicas de amostragem e quantização em sinais analógicos, obtendo sinais digitais, processo conhecido como conversão analógico-digital (CAD), faz-se necessária para que o estudo de tais objetos seja possível.

2.2. Amostragem

O estudo de sinais de áudio em sua forma analógica, ou seja, representados através de sinais contínuos ao longo do tempo, trata-se de uma tarefa muito difícil. Logo, uma abordagem discreta acerca do estudo de sinais analógicos se faz necessária.

A técnica de amostragem é responsável por representar um sinal analógico como um conjunto de valores discretos, ainda tomando a variável temporal como referência [9].

[2] Estatística de inscritos do concurso público estadual para o cargo de perito criminal da superintendência da polícia técnico-científica do estado de São Paulo no ano de 2013, fundação VUNESP, disponível em http://www.vunesp.com.br/PCSP1302/PCSP1302_310_011645.pdf. Acessado em 24/01/2019.

Tais elementos discretos são números que representam valores de amplitude do sinal em intervalos de tempo regulares de tempo. O processo de amostragem é ilustrado na Fig. 1.

Existem diversas possibilidades para a digitalização de sinais analógicos. Entretanto, a representação mais utilizada para este fim, na área de PDS, é a chamada modulação por código de pulso (PCM, *Pulse code modulation*) [10]. Tal representação baseia-se na amostragem de um sinal analógico em termos de sequências de números binários, os quais são mais facilmente manipulados e armazenados em sistemas computacionais.

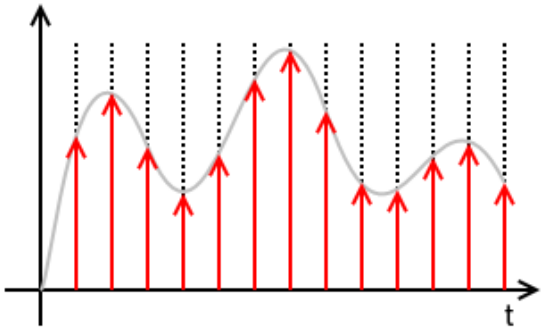

Figura 1. Amostragem de um sinal contínuo.

2.3. Quantização

As técnicas de amostragem representam sinais analógicos discretamente por meio de valores numéricos, os quais representam a amplitude do sinal, em intervalos regulares de tempo. No entanto, sabe-se que, somente com técnicas de amostragem, distâncias demasiadamente grandes entre os valores contínuos e os discretos devem ser observadas.

Assim, a técnica de quantização surge como alternativa para minimizar a distância entre os valores contínuos e discretos no processo de digitalização de um sinal analógico [9]. A Fig. 2 ilustra a diferença um sinal digital quantizado.

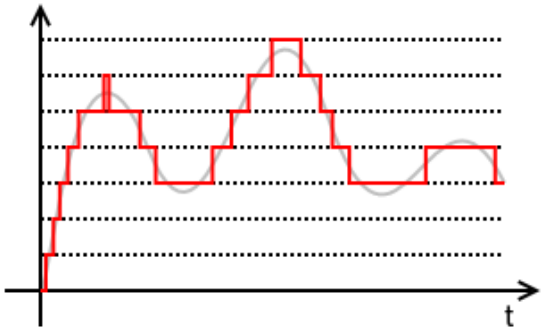

Figura 2. Sinal digital quantizado.

2.4. Processamento de sinais de voz: os conceitos de ZCR e de energia

Sinais analógicos de áudio devidamente digitalizados e quantizados, representados por meio de vetores numéricos, invariavelmente, possuem comprimentos distintos. Tal conclusão resulta do fato de que, mesmo que determinado locutor grave inúmeras amostras de voz pronunciando o mesmo conteúdo fonético, nenhuma amostra será idêntica em relação à outra.

Assim, técnicas de processamento digital de áudio se fazem necessárias para o tratamento de tais sinais digitais, de modo que os referidos vetores numéricos possam ser facilmente manipulados padronizando-os no que diz respeito seus tamanhos, logo, temos como resultado os chamados vetores de característica oriundos de um procedimento conhecido como extração de características [9].

Dentre as inúmeras técnicas utilizadas na área de PDS, podemos citar critérios como: balanceamento energético, magnitude, centroide espectral, taxa de passagem pelo zero (ZCR, *Zero-rossing rate*), entre outros [11,12].

O critério de balanceamento energético o longo do tempo e da frequência relaciona-se diretamente aos valores máximos e mínimos das frequências de um sinal. Tal critério torna-se efetivo no que consiste na distinção entre vozes de sexo feminino e do sexo masculino, sendo apropriado para situações relacionadas ao reconhecimento biométrico por voz [1]. O valor de energia, E, de um dado sinal digital é calculado através da Eq. (1):

$$E = \sum_{i=-\infty}^{\infty} x^2[i], \tag{1}$$

onde $x[i]$ é a amplitude do sinal na posição i.

Já o critério ZCR baseia-se quantidade de passagens pelo zero do sinal analisado. Tal critério, em combinação com o discutido critério de balanceamento energético, adequa-se a situações de identificação biométrica por voz uma vez que proporciona uma análise combinada tempo-frequencia e auxilia na detecção de sinais audíveis, neste caso, apresentando alto balanço energético e baixa taxa de passagem pelo zero [12].

3. METODOLOGIA

O presente capítulo descreve a metodologia adotada na execução do trabalho e é dividido em três seções. A primeira seção, 3.1, descreve os procedimentos inerentes às coletas das amostras de voz para formação da base de

dados utilizada no decorrer do trabalho. A seção 3.2 relata a manipulação dos arquivos de áudio, obtidos na etapa anterior, para a extração de características dos mesmos. Por fim, na terceira e última etapa, seção 3.3, é projetado e implementado os mecanismos necessários para o reconhecimento biométrico por voz.

3.1. Formação da base de dados

A formação da base de dados, constituída por arquivos de áudio em formato *wave*, deu-se através da gravação de amostras de voz divididas entre 12 (doze) locutores distintos, sendo 7 (sete) locutores do sexo feminino e 5 (cinco) locutores do sexo masculino de diferentes faixas etárias. Vale ressaltar que o formato *wave*, *Waveform Audio File Format*, cuja extensão é *.wav*, desenvolvido pela empresa *Microsoft* em conjunto com a IBM, foi escolhido como extensão para as coletas de voz por caracterizar-se como um formato sem compressão, ou seja, é formado, basicamente, por um cabeçalho e dados brutos.

Para cada locutor foram gravados 5 (cinco) arquivos de áudio, totalizando $(12\ \text{locutores}) \times (5\ \text{amostras}) = 60\ \text{sinais}$, utilizando como conteúdo fonético a frase pré-determinada *"A justiça é cega"*.

O *software* utilizado para a obtenção dos arquivos de áudio foi o *Audacity*[3], um *software* gratuito e de fácil manuseio. Cada um dos 60 (sessenta) arquivos de áudio foi gravado com taxa de amostragem maior ou igual a 8000 amostras/seg e quantização de 16 bits, sendo que os mesmos foram editados através do próprio *Audacity* de forma que seus inícios e seus finais fossem removidos. Tal medida foi necessária para que os arquivos de áudio fossem preenchidos apenas com as vozes de seus respectivos locutores.

3.2. Extração de características

O objetivo desta etapa foi transformar discretamente cada arquivo de áudio obtido na etapa anterior em vetores de tamanhos fixos, uma vez que originalmente cada um dos arquivos de áudio da base de dados possuía tamanhos distintos entre si. Para isso, efetuamos a extração de certas características dos arquivos de áudio, de forma a gerar vetores com o número de posições fixas, vetores de característica.

Optou-se para a extração de características utilizando dois diferentes critérios, os quais foram utilizados isolados e combinados. Os critérios são a extração de parâmetros relativos às concentrações energéticas ao longo do tempo e da frequência e o critério ZCR. Definimos 8 (oito) diferentes possibilidades de extração de características, as quais possibilitaram 8 (oito) modelos distintos para a classificação biométrica por voz. Tais possibilidades são:

Modelo 1: Extração de características baseado no critério das concentrações energéticas, gerando um vetor com 9 (nove) posições.

Modelo 2: Extração de características baseado no critério ZCR, gerando um vetor com 9 (nove) posições.

Modelo 3: Extração de características baseado no critério das concentrações energéticas, gerando um vetor com 19 (dezenove) posições.

Modelo 4: Extração de características baseado no critério ZCR, gerando um vetor com 19 (dezenove) posições.

Modelo 5: Extração de características baseado na utilização de ambos os critérios, gerando um vetor com 18 (dezoito) posições. Neste caso as 9 (nove) primeiras posições são oriundas do critério das concentrações energéticas e as demais 9 (nove) posições oriundas do critério ZCR.

Modelo 6: Extração de características baseado na utilização de ambos os critérios, gerando um vetor com 38 (trinta e oito) posições. Neste caso as 19 (dezenove) primeiras posições são oriundas do critério das concentrações energéticas e as demais 19 (dezenove) posições oriundas do critério ZCR.

Modelo 7: Extração de características baseado na utilização de ambos os critérios, gerando um vetor com 28 (vinte e oito) posições. Neste caso as 9 (nove) primeiras posições são oriundas do critério das concentrações energéticas e as demais 19 (dezenove) posições oriundas do critério ZCR.

Modelo 8: Extração de características baseado na utilização de ambos os critérios, gerando um vetor com 28 (vinte e oito) posições. Neste caso as 19 (dezenove) primeiras posições são oriundas do critério das concentrações energéticas e as demais 9 (nove) posições oriundas do critério ZCR.

3.2.1. Algoritmo – Extração de características

Início;

Passo 1) Entrada de dados: Vetor de dados brutos e seu respectivo tamanho;

Passo 2) Obter a média dos valores que formam o vetor de dados brutos;

Passo 3) Transladar o vetor de dados brutos subtraindo a média obtida no passo 2 em cada uma de suas posições;

Passo 4) Calcular/acumular a quantidade de cruzamentos por zero (critério ZCR);

Passo 5) Calcular a energia total do vetor de dados brutos;

Passo 6) Calcular as posições do vetor de dados brutos que correspondem a porcentagens da energia total calculada no passo 5, dividir pelo tamanho total do vetor de dados brutos e armazenar os valores no vetor de característica;

Passo 7) Calcular as posições do vetor de dados brutos que correspondem a porcentagens do ZCR total calculada no passo 4, dividir pelo tamanho total do vetor de dados brutos e armazenar os valores no vetor de característica;

Passo 8) Imprimir todas as posições do vetor de característica;

Fim;

3.3. Projeto e implementação do classificador

Para o planejamento de um classificador adequado, inicialmente, para cada locutor, os vetores de característica obtidos dos 60 (sessenta) sinais foram representados graficamente por meio do *software* livre *Writer* (aplicativo pertencente ao pacote *LibreOffice*).

Os gráficos relativos a cada locutor foram sobrepostos, possibilitando a verificação da similaridade entre as curvas obtidas para um mesmo locutor e diferenças consideráveis entre as curvas de locutores distintos. Por meio da análise anterior concluiu-se que um classificador baseado em uma distribuição Gaussiana seria suficiente para a obtenção de bons resultados, a qual está definida na Eq. (2):

$$Locutor_k = \exp\left[-\frac{\sum_{i=1}^{j} \left(x[i] - \mu_k[i] \right)^2}{2\sum_{i=1}^{j} \sigma_k^2[i]} \right], \quad (2)$$

onde $x[i]$ corresponde a posição i do vetor de entrada, $\mu_k[i]$ corresponde a posição de média i referente ao locutor k e $\sigma_k[i]$ corresponde a posição de variância i do locutor k.

Para a devida utilização da Eq. (2), obtemos valores estatísticos de média e variância para o conjunto de 5 (cinco) vetores de característica de cada um dos locutores. Neste caso, foram utilizadas funções próprias do *Writer* para obter tais valores de estatísticos.

3.3.1. Algoritmo – Classificador

Início;

Passo 1) Entrada de dados: Posições do vetor de característica do arquivo de áudio analisado;

Passo 2) Com base no vetor de característica (entrada) e nos valores estatísticos, calcular as probabilidades de correspondência, utilizando a distribuição gaussiana, para cada um dos locutores da base de dados;

Passo 3) Guardar o índice que corresponde ao locutor mais provável;

Passo 4) Imprimir o índice do locutor mais provável com seu respectivo valor de probabilidade;

Fim;

4. RESULTADOS

Devido a metodologia apresentar diversos passos de análise, por simplicidade, são apresentados apenas os resultados acerca do modelo que mais se adequou ao problema proposto pelo trabalho, entretanto, é válido ressaltar que foram testados 8 (oito) modelos distintos, sendo que os procedimentos aqui apresentados foram repetidos por mais 7 (sete) vezes.

Foram testados 8 (oito) possibilidades para o vetor de característica, gerando 8 (oito) modelos distintos. Entretanto, o modelo mais adequado, apresentando porcentagem de acertos de 70%, foi o modelo que teve o vetor de característica composto por 19 (dezenove) posições oriundas apenas do critério ZCR.

Logo, após a devida coleta e formação da base de dados, o algoritmo desenvolvido para extração de característica foi utilizado gerando vetores de característica para cada amostra de voz, os quais, para cada locutor, foram graficados e estão apresentados na Fig. 3.

Analisando os gráficos da Fig. 3 é possível observar diferenças no comportamento da curva para cada locutor. Assim, concluímos que, como discutido no capítulo anterior, a utilização da distribuição Gaussiana de probabilidades seria suficiente para classificar efetuar o trabalho de classificação entre tais indivíduos.

Entretanto, antes de utilizar a distribuição gaussiana para efetuar a classificação, Eq. (2), obtemos valores esta-

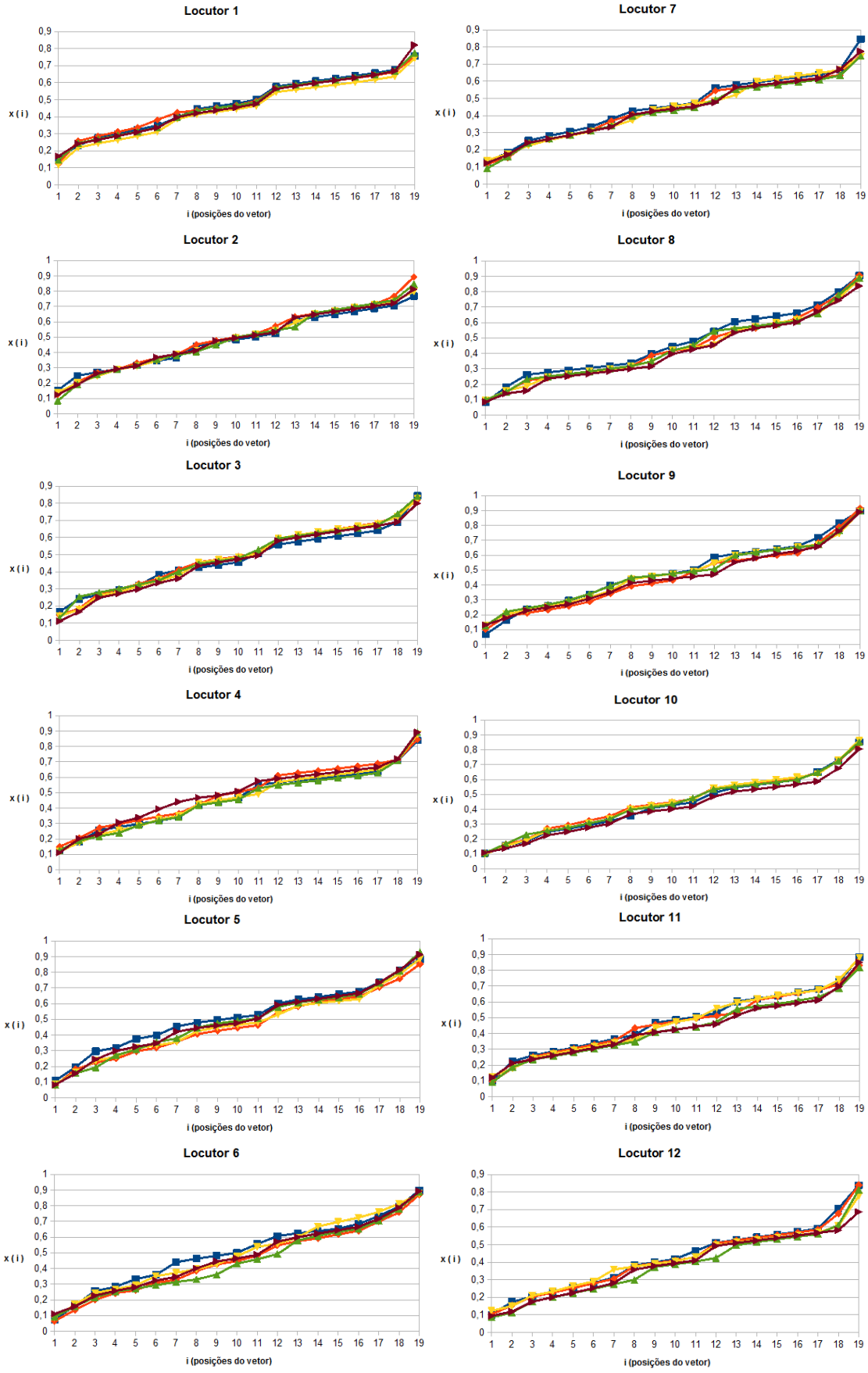

Figura 3. Gráficos: vetor de característica em função da posição.

tísticos de média e variância para cada um dos 12 (doze) locutores. Para fins de ilustração, a Fig. 4 expõe os gráficos da média, μ, e da variância, σ^2, para o locutor 1. Vale ressaltar que a escolha do locutor 1 foi feita de maneira aleatória, sem fins específicos.

Colocando os valores de média e variância para todos os locutores da base de dados no algoritmo classificador foi possível testar cada um dos 60 (sessenta) sinais coletados, de forma que os resultados obtidos estão expressos na seguinte matriz de confusão, onde, para melhor interpretação da mesma, valores no decorrer das linhas devem indicar o que deve ocorrer idealmente e valores ao longo das colunas devem indicar os resultados obtidos por meio do classificador.

	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12
L1	4	0	0	0	0	0	0	0	0	0	0	0
L2	0	4	0	1	0	0	0	0	0	0	0	0
L3	0	0	3	0	0	0	0	0	0	0	0	0
L4	0	0	2	3	0	0	0	0	0	0	0	0
L5	0	1	0	0	3	1	0	0	1	0	0	0
L6	0	0	0	0	2	3	0	1	0	0	2	0
L7	1	0	0	0	0	0	4	0	0	0	0	0
L8	0	0	0	0	0	1	0	4	0	0	0	0
L9	0	0	0	0	0	0	0	0	3	0	0	0
L10	0	0	0	0	0	0	0	0	0	3	0	0
L11	0	0	0	1	0	0	1	0	1	1	3	0
L12	0	0	0	0	0	0	0	0	0	1	0	5

Portanto, analisando a matriz de confusão representada, a diagonal principal representa a quantidade de acertos obtidos pelo classificador implementado. Assim, do total de 60 (sessenta) sinais analisados, o classificador obteve 42 (quarenta e dois) acertos, indicando uma porcentagem de acertos de 70%.

Por fim, a Tab. 1 indica as porcentagens de acerto obtidas por todos os modelos testados. Percebe-se, pela análise da Tab.1, acurácias na faixa de 41,67% até 70%. No primeiro caso, a acurácia teve como base uma análise por vetores de características compostos por 9 posições de energia, implicando em uma análise de cunho temporal e sem detalhamento. Por outro lado, no segundo caso, a acurácia teve por base uma análise mais refinada, contando com 19 posições do critério ZCR. Assim, entende-se que a análise em frequência, oriunda dos ZCRs, foi determinante. Esse resultado condiz com aqueles encontrados na literatura da área [1].

5. CONCLUSÕES

Após o estudo, projeto e a implementação do algoritmo classificador biométrico baseado no padrão acústico de locutores, o **modelo 4**, baseado no vetor de característica formado por 19 (dezenove) posições oriundas do critério ZCR, apresentou os melhores resultados no que se refere aos objetivos propostos pelo presente trabalho.

Tabela 1. Resultados obtidos pelo classificador implementado.

Critério/Tamanho do vetor de característica	Porcentagem de acertos (acurácia)
ZCR/19 posições	70,00%
ZCR/9 posições	63,33%
Energia e ZCR/28 posições (9E e 19ZCR)	56,67%
Energia e ZCR/38 posições (19E e 19ZCR)	55,00%
Energia e ZCR/18 posições (9E e 9ZCR)	55,00%
Energia e ZCR/28 posições (19E e 9ZCR)	50,00%
Energia/19 posições	43,33%
Energia/9 posições	41,67%

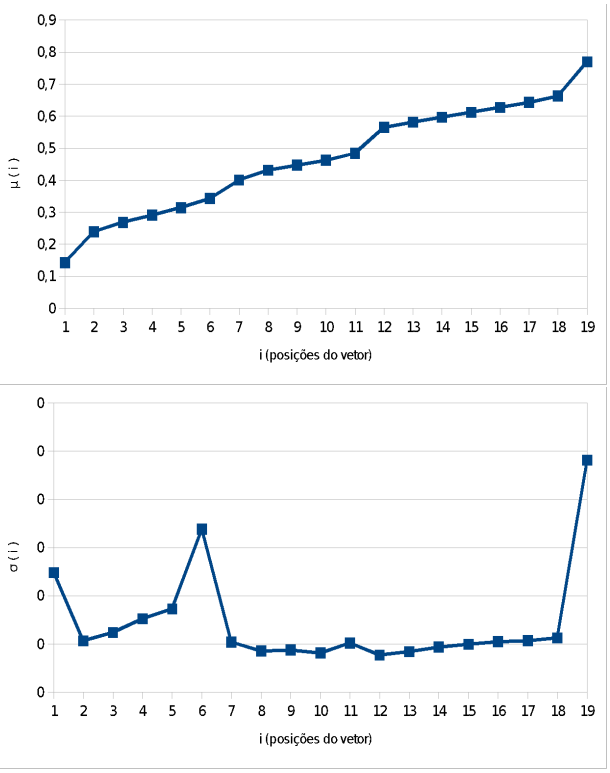

Figura 4. Gráficos dos vetores média (superior) e variância (inferior), ambos para o locutor 1.

Tal modelo apresentou uma margem de acerto de 70%, caracterizando-se como condizente, se considerarmos que a base de dados coletada para a execução do trabalho for considerada demasiadamente pequena. Para que a aplicação desenvolvida se torne comercial, além do desenvolvimento de uma interface gráfica para facilitar a interação do usuário, são necessários testes considerando bases de dados com milhares de locutores.

Logo, dentro de suas limitações, a abordagem utilizada no decorrer do projeto mostrou-se eficiente, no que se refere à identificação biométrica de indivíduos com base em seus padrões acústicos.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] H. Beigi. *Fundamentals of speaker recognition*. Yorktown: Springer (2011).
[2] D. A. Armiato. Reconhecimento de fala com vocabulário limitado baseado em descritores energético-entrópicos e densidade gaussiana. *Trabalho de Conclusão de Curso*, Departamento de Ciência de Computação e Estatística, Universidade Estadual "Júlio de Mesquita Filho" (2012).
[3] R. Trevizam. Um algoritmo para reconhecimento de locutores baseado na transformada discreta de Haar. *Trabalho de Conclusão de Curso*, Departamento de Ciência de Computação e Estatística, Universidade Estadual "Júlio de Mesquita Filho" (2015).
[4] C. H. Calazans; S. M. Calazans. Ciência forense: das origens à ciência forense computacional. *XV Seminário Regional de Informática*, Universidade Integrada, Santo Ângelo/RS (2005).
[5] J. A. Velho; G. C. Geiser; A. Espíndula. Ciências forenses: Uma introdução às principais áreas da criminalística moderna. Campinas: Millennium (2013).
[6] C. Lenke. Importância da ciência forense computacional. Artigo publicado no portal *under-linux.org* em 01/06/2013, disponível em https://under-linux.org/content.php?r=6665. Acessado em 12/01/2019.
[7] R. N. Almeida. Perícia forense computacional: Estudo das técnicas utilizadas para coleta e análise de vestígios digitais. *Trabalho de Conclusão de Curso*, FATEC/SP (2011).
[8] C. S. Yoshizaki. Análise de 15 STRS autossômicos na população de Araraquara (São Paulo, Brasil). *Trabalho de Conclusão de Curso*, Faculdade de Ciências Farmacêuticas/UNESP (2011).
[9] S. Haykin; B. Veen. Sinais e sistemas. Porto Alegre: Bookman (2002).
[10] K. W. Cattermole. *Principles of pulse code modulation*. American Elsevier Pub. Co (1989).
[11] M. Al-akaidi. *Fractal speech processing*. New York: Cambridge University Press (2004).
[12] R. C. Guido. *ZCR-aided Neurocomputing: a study with applications*. Knowledge-based Systems, v. 105, pp. 248 – 269 (2016).

APÊNDICE – CÓDIGOS FONTE

A seguir estão expostos, para fins de auxílio para o entendimento, os códigos fonte completos, de acordo com o **modelo 5**, dos algoritmos utilizados para:

(a) Extração de características;

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<iostream>
//-----------------------------------
int main(int i,char* n[])
{
void
analisa_dados_brutos(double*,int);
short converte2de8para1de16(unsigned
char, unsigned char);
FILE* fr;

if(((fr=fopen(n[1],"rb"))!=NULL))
        {
        struct
                {
                unsigned char riff[4];
                unsigned int len;
                } riff_header;

fread(&riff_header,sizeof(riff_header)
,1,fr);
        std::cout<<"\nArquivo do tipo:
"<<riff_header.riff[0]<<riff_header.ri
ff[1]<<riff_header.riff[2]<<riff_heade
r.riff[3];
        std::cout<<"\nTamanho
excluindo header: "<<riff_header.len;

//////////////////////////////////////

        unsigned char wave[4];

fread(&wave,sizeof(wave),1,fr);
        std::cout<<"\nSub-Tipo:
"<<wave[0]<<wave[1]<<wave[2]<<wave[3];

//////////////////////////////////////
        struct
                {
                unsigned char id[4];
                unsigned int len;
                } riff_chunk;

fread(&riff_chunk,sizeof(riff_chunk),1
,fr);
        std::cout<<"\nIdentificador:
"<<riff_chunk.id[0]<<riff_chunk.id[1]<
<riff_chunk.id[2]<<riff_chunk.id[3];
```

```
        std::cout<<"\nComprimento do
chunk apos header: "<<riff_chunk.len;



        struct
                {
                unsigned short
formattag;
                unsigned short
numberofchannels;
                unsigned int
samplingrate;
                unsigned int
avgbytespersecond;
                unsigned short
blockalign;
                }  wave_chunk;

fread(&wave_chunk,sizeof(wave_chunk),1
,fr);

//tratamento de uma excessao que
costuma aparecer em alguns arquivos
wav... O correto seriam 16 bytes, as
vezes aparecem 18 ou mais...
      if(riff_chunk.len>16)
                  {
                  unsigned char
excesso;
                for(int
i=0;i<riff_chunk.len-16;i++)

      fread(&excesso,sizeof(excesso),1,
fr);
                  }
      //fim do tratamento da excess?
        std::cout<<"\nCategoria do
formato: "<<wave_chunk.formattag;
        std::cout<<"\nNumero de
canais:
"<<wave_chunk.numberofchannels;
        std::cout<<"\nTaxa de
amostragem:
"<<wave_chunk.samplingrate;
        std::cout<<"\nMedia do num. de
bps: "<<wave_chunk.avgbytespersecond;
        std::cout<<"\nAlinhamento do
bloco em bytes:
"<<wave_chunk.blockalign;

////////////////////////////////////

        if(wave_chunk.formattag==1)
//PCM
                {
                int
resolucao=(wave_chunk.avgbytespersecon
d * 8)/(wave_chunk.numberofchannels *
wave_chunk.samplingrate);// pq nao
bitssample

std::cout<<"\nResolucao: "<<resolucao;
                struct
                        {
                        unsigned char
data[4];
                        unsigned int
chunk_size;
                        }
header_data_chunk;


fread(&header_data_chunk,sizeof(header
_data_chunk),1,fr);

std::cout<<"\nIdentificacao:
"<<header_data_chunk.data[0]<<header_d
ata_chunk.data[1]<<header_data_chunk.d
ata[2]<<header_data_chunk.data[3];
                std::cout<<"\nTamanho
do chunk de dados:
"<<header_data_chunk.chunk_size;
                std::cout<<"\nNumero
de frames para amostrar:
"<<header_data_chunk.chunk_size/wave_c
hunk.blockalign;

                int
tamanho_da_janela=header_data_chunk.ch
unk_size/wave_chunk.blockalign;

                std::cout<<"\nTamanho
da janela: "<<tamanho_da_janela;
                if((resolucao==8) &&
(wave_chunk.numberofchannels==1))
                        {
                         unsigned char
waveformdata;
                         double*
amostras_no_tempo = new
double[tamanho_da_janela];
                         for(int
i=0;i<tamanho_da_janela;i++)
                                 {

fread(&waveformdata,sizeof(waveformdat
a),1,fr);

amostras_no_tempo[i]=(double)waveformd
ata;
                                 }


      analisa_dados_brutos(&amostras_no
_tempo[0],tamanho_da_janela);
                        }
                else if((resolucao==8)
&& (wave_chunk.numberofchannels==2))
                        {
                         unsigned char
waveformdata_right;
                         unsigned char
waveformdata_left;
                         double*
amostras_no_tempo_left = new
double[tamanho_da_janela];
```

```
                        double*
amostras_no_tempo_right = new
double[tamanho_da_janela];
                        for(int
i=0;i<tamanho_da_janela;i++)
                                {

fread(&waveformdata_left,sizeof(wavefo
rmdata_left),1,fr);

fread(&waveformdata_right,sizeof(wavef
ormdata_right),1,fr);


amostras_no_tempo_right[i]=(double)wav
eformdata_right;

amostras_no_tempo_left[i]=(double)wave
formdata_left;
                                }


      analisa_dados_brutos(&amostras_no
_tempo_left[0],tamanho_da_janela);
                                //
      analisa_dados_brutos(&amostras_no
_tempo_right[0],tamanho_da_janela);
                        }
                else
if((resolucao==16) &&
(wave_chunk.numberofchannels==1))
                        {
                        unsigned char
waveformdata_lsb, waveformdata_msb;
                        double*
amostras_no_tempo = new
double[tamanho_da_janela];
                        for(int
i=0;i<tamanho_da_janela;i++)
                                {

fread(&waveformdata_lsb,sizeof(wavefor
mdata_lsb),1,fr);

fread(&waveformdata_msb,sizeof(wavefor
mdata_msb),1,fr);

amostras_no_tempo[i]=(double)converte2
de8para1de16(waveformdata_lsb,waveform
data_msb);
                                }


      analisa_dados_brutos(&amostras_no
_tempo[0],tamanho_da_janela);
                        }
                else if
((resolucao==16) &&
(wave_chunk.numberofchannels==2))
                        {
                        unsigned char
waveformdata_lsb_left,
waveformdata_lsb_right,
waveformdata_msb_left,
waveformdata_msb_right;
                        double*
amostras_no_tempo_left = new
double[tamanho_da_janela];
                        double*
amostras_no_tempo_right = new
double[tamanho_da_janela];
                        for(int
i=0;i<tamanho_da_janela;i++)
                                {

fread(&waveformdata_lsb_left,sizeof(wa
veformdata_lsb_left),1,fr);

fread(&waveformdata_msb_left,sizeof(wa
veformdata_msb_left),1,fr);

fread(&waveformdata_lsb_right,sizeof(w
aveformdata_lsb_right),1,fr);

fread(&waveformdata_msb_right,sizeof(w
aveformdata_msb_right),1,fr);

amostras_no_tempo_left[i]=(double)conv
erte2de8para1de16(waveformdata_lsb_lef
t,waveformdata_msb_left);

amostras_no_tempo_right[i]=(double)con
verte2de8para1de16(waveformdata_lsb_ri
ght,waveformdata_msb_right);
                                }


      analisa_dados_brutos(&amostras_no
_tempo_left[0],tamanho_da_janela);
                                //
      analisa_dados_brutos(&amostras_no
_tempo_right[0],tamanho_da_janela);
                        }
                else

      std::cout<<"Resolucao ou nmero de
canais invalido(s)";

                }
        else
                std::cout<<"FORA DO
FORMATO PCM...";
        fclose(fr);
        }
else
        std::cout<<"Arquivo nao existe
ou nao pode ser aberto";
std::cout<<"\n\n\n";
}
//-------------------------------------
short converte2de8para1de16(unsigned
char lsb, unsigned char msb)
{
return(((msb&0x80)>>7)*(32768) +
       ((msb&0x40)>>6)*(16384) +
       ((msb&0x20)>>5)*(8192) +
       ((msb&0x10)>>4)*(4096) +
```

```
       ((msb&0x08)>>3)*(2048) +
       ((msb&0x04)>>2)*(1024) +
       ((msb&0x02)>>1)*(512) +
       ((msb&0x01))*(256) +
       ((lsb&0x80)>>7)*(128) +
       ((lsb&0x40)>>6)*(64) +
       ((lsb&0x20)>>5)*(32) +
       ((lsb&0x10)>>4)*(16) +
       ((lsb&0x08)>>3)*(8) +
       ((lsb&0x04)>>2)*(4) +
       ((lsb&0x02)>>1)*(2) +
       (lsb&0x01));
}
//-------------------------------------
void analisa_dados_brutos(double* s,
int N)
{
//Declaração das variáveis
double
en_total,vet_carac[18],en1,en2,zcr1,zc
r2,media;
int i,j,zcr;

//Obter a média dos valores do sinal
wav
media=0;
for(i=0;i<N;i++)
      media=media+s[i];

media=media/N;

//Translação do sinal wav
for(i=0;i<N;i++)
      s[i]=s[i]-media;

//Cálculo do ZCR, ou seja, quantidade
de cruzamentos por 0 ocorridos
zcr=0;
for(i=0;i<(N-1);i++)
      if(s[i]*s[i+1]<0)
            zcr++;

printf("\n ZCR = %i \n",zcr);

//Laço para calcular a energia total
do sinal wav
en_total=0;
for(i=0;i<N;i++)
      en_total=en_total+s[i]*s[i];

printf("\n Energia Total = %f
\n",en_total);

//Laço para zerar todas as posições do
vetor de características
for(i=0;i<18;i++)
      vet_carac[i]=0;

//Laço para calcular as posições
relativas a energia do vetor de
características
for(i=0;i<9;i++){
      en1=0;
      en2=(i+1)*(0.1)*en_total;
      j=0;
      while(en1 < en2){
                  en1=en1+s[j]*s[j];
                  j++;
               }

      vet_carac[i]=((double)(j))/((doub
le)(N));
}

//Laço para calcular as posições
relativas ao ZCR do vetor de
características
for(i=0;i<9;i++){
      zcr1=0;
      zcr2=(i+1)*(0.1)*zcr;
      j=0;
      while(zcr1 < zcr2){
            if(s[j]*s[j+1]<0)
                  zcr1++;

            j++;}

      vet_carac[i+9]=((double)(j))/((do
uble)(N));
}
//Impressão do vetor de
características
printf("\n Vetor de Caracteristicas
\n");
for(i=0;i<18;i++)
      printf("%f;",vet_carac[i]);
}
//-------------------------------------
```

(b) Classificação de locutores;

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<iostream>
//-------------------------------------
using namespace std;

double dist2(double s[],double
m[][18],int n){
//Declaração de variáveis
double valor=0;

int i;

for(i=0;i<18;i++)
      valor=valor+(s[i]-m[n][i])*(s[i]-
m[n][i]);

return valor;}

double modulo(double v[][18],int n){
//Declaração de variáveis
double valor=0;

int i;

for(i=0;i<18;i++)
```

```
      valor=valor+v[n][i]*v[n][i];

valor=sqrt(valor);

return valor;}

int main(void){
//Declaração das variáveis
double
med[12][18],var[12][18],sinal[18],mode
lo[12],conv;

int i,indice,j;

//Valores de média e variância
//Locutor1
med[0][0]=0.071947;
      var[0][0]=0.0002272483;
med[0][1]=0.117256;
      var[0][1]=0.0007878334;
med[0][2]=0.1855912;
      var[0][2]=0.0018819862;
med[0][3]=0.2764862;
      var[0][3]=0.000666779;
med[0][4]=0.340553;
      var[0][4]=0.0009399396;
med[0][5]=0.4440972;
      var[0][5]=0.0016825944;
med[0][6]=0.5391948;
      var[0][6]=0.0010246296;
med[0][7]=0.5997232;
      var[0][7]=0.0002498665;
med[0][8]=0.6721836;
      var[0][8]=0.0013696233;
med[0][9]=0.2390886;
      var[0][9]=0.0002126568;
med[0][10]=0.290815;
      var[0][10]=0.0003043483;
med[0][11]=0.3428504;
      var[0][11]=0.0006764106;
med[0][12]=0.4313066;
      var[0][12]=0.0001707283;
med[0][13]=0.4622416;
      var[0][13]=0.0001627222;
med[0][14]=0.564925;
      var[0][14]=0.0001534491;
med[0][15]=0.5967372;
      var[0][15]=0.0001871797;
med[0][16]=0.626733;
      var[0][16]=0.0002096173;
med[0][17]=0.662825;
      var[0][17]=0.0002248687;

//Locutor2
med[1][0]=0.0893492;
      var[1][0]=0.0005213028;
med[1][1]=0.1657362;
      var[1][1]=0.016117171;
med[1][2]=0.2727984;
      var[1][2]=0.0320167051;
med[1][3]=0.4092156;
      var[1][3]=0.0446592919;
med[1][4]=0.5467278;
      var[1][4]=0.0032209527;
med[1][5]=0.5993022;
      var[1][5]=0.0007261089;
med[1][6]=0.6476484;
      var[1][6]=0.0045222905;
med[1][7]=0.7444566;
      var[1][7]=0.0023673469;
med[1][8]=0.7974042;
      var[1][8]=0.0002089941;
med[1][9]=0.2080212;
      var[1][9]=0.0005676299;
med[1][10]=0.2906444;
      var[1][10]=0.000001532;
med[1][11]=0.355799;
      var[1][11]=0.0001069557;
med[1][12]=0.4249802;
      var[1][12]=0.0004193731;
med[1][13]=0.4948806;
      var[1][13]=0.0000412893088;
med[1][14]=0.5431196;
      var[1][14]=0.0003059832;
med[1][15]=0.648372;
      var[1][15]=0.0001137375;
med[1][16]=0.6879436;
      var[1][16]=0.000142845;
med[1][17]=0.7341898;
      var[1][17]=0.0005222968;

//Locutor3
med[2][0]=0.0644104;
      var[2][0]=0.0002533825;
med[2][1]=0.0921454;
      var[2][1]=0.0003421279;
med[2][2]=0.1607096;
      var[2][2]=0.0010361317;
med[2][3]=0.2297802;
      var[2][3]=0.0015341019;
med[2][4]=0.3325206;
      var[2][4]=0.005646615;
med[2][5]=0.4421986;
      var[2][5]=0.0032900986;
med[2][6]=0.5328294;
      var[2][6]=0.0003455516;
med[2][7]=0.6603518;
      var[2][7]=0.0045152211;
med[2][8]=0.7471812;
      var[2][8]=0.0006367461;
med[2][9]=0.2058688;
      var[2][9]=0.0015819441;
med[2][10]=0.2899026;
      var[2][10]=0.0001422195;
med[2][11]=0.3559676;
      var[2][11]=0.0003444151;
med[2][12]=0.4430992;
      var[2][12]=0.000170811;
med[2][13]=0.4771824;
      var[2][13]=0.0001797253;
med[2][14]=0.5841978;
      var[2][14]=0.0002177205;
med[2][15]=0.6207474;
      var[2][15]=0.000265167;
med[2][16]=0.6531502;
      var[2][16]=0.0003021235;
med[2][17]=0.7134054;
      var[2][17]=0.0004621227;
```

```
//Locutor4
med[3][0]=0.1091442;
      var[3][0]=0.0014051251;
med[3][1]=0.1784756;
      var[3][1]=0.0030649314;
med[3][2]=0.2549288;
      var[3][2]=0.005223712;
med[3][3]=0.3498934;
      var[3][3]=0.0057721124;
med[3][4]=0.451305;
      var[3][4]=0.0069325028;
med[3][5]=0.5352586;
      var[3][5]=0.0136791599;
med[3][6]=0.6470608;
      var[3][6]=0.0104448396;
med[3][7]=0.7528086;
      var[3][7]=0.0044125248;
med[3][8]=0.8202144;
      var[3][8]=0.0038102823;
med[3][9]=0.191536;
      var[3][9]=0.0001470968;
med[3][10]=0.2755964;
      var[3][10]=0.000763491;
med[3][11]=0.3408498;
      var[3][11]=0.0009881686;
med[3][12]=0.431528;
      var[3][12]=0.000399789;
med[3][13]=0.4812462;
      var[3][13]=0.0005620267;
med[3][14]=0.5773594;
      var[3][14]=0.0005891581;
med[3][15]=0.607698;
      var[3][15]=0.0005891878;
med[3][16]=0.6373956;
      var[3][16]=0.0005901877;
med[3][17]=0.711687;
      var[3][17]=0.0000178051464999999;

//Locutor5
med[4][0]=0.0709424;
      var[4][0]=0.0006147719;
med[4][1]=0.0901534;
      var[4][1]=0.0005842862;
med[4][2]=0.1138896;
      var[4][2]=0.0004649349;
med[4][3]=0.18554;
      var[4][3]=0.0041890916;
med[4][4]=0.3207764;
      var[4][4]=0.0130580579;
med[4][5]=0.5033006;
      var[4][5]=0.005459016;
med[4][6]=0.6105696;
      var[4][6]=0.0039829564;
med[4][7]=0.7079018;
      var[4][7]=0.0001798259;
med[4][8]=0.74024;
      var[4][8]=0.0003166876;
med[4][9]=0.1700504;
      var[4][9]=0.0003095356;
med[4][10]=0.2801416;
      var[4][10]=0.0008210037;
med[4][11]=0.3507022;
      var[4][11]=0.0009081113;
med[4][12]=0.4401968;
      var[4][12]=0.0007496002;
med[4][13]=0.4778376;
      var[4][13]=0.0006332014;
med[4][14]=0.5692946;
      var[4][14]=0.0009517865;
med[4][15]=0.622943;
      var[4][15]=0.0002265036;
med[4][16]=0.6555182;
      var[4][16]=0.0003522587;
med[4][17]=0.7968704;
      var[4][17]=0.0005423184;

//Locutor6
med[5][0]=0.085575;
      var[5][0]=0.0023860329;
med[5][1]=0.1277122;
      var[5][1]=0.0049362431;
med[5][2]=0.2121574;
      var[5][2]=0.0159215527;
med[5][3]=0.3142974;
      var[5][3]=0.0200474715;
med[5][4]=0.43366;
      var[5][4]=0.020253463;
med[5][5]=0.5468066;
      var[5][5]=0.0146179266;
med[5][6]=0.6541394;
      var[5][6]=0.0130454996;
med[5][7]=0.7385806;
      var[5][7]=0.0024557458;
med[5][8]=0.791603;
      var[5][8]=0.0012067111;
med[5][9]=0.1570814;
      var[5][9]=0.0001754376;
med[5][10]=0.260988;
      var[5][10]=0.0002269145;
med[5][11]=0.3272966;
      var[5][11]=0.0007566678;
med[5][12]=0.3946142;
      var[5][12]=0.0022255402;
med[5][13]=0.4636654;
      var[5][13]=0.0007111742;
med[5][14]=0.5549238;
      var[5][14]=0.0017391313;
med[5][15]=0.6261202;
      var[5][15]=0.0008521935;
med[5][16]=0.6735308;
      var[5][16]=0.0010767431;
med[5][17]=0.7871116;
      var[5][17]=0.0004233264;

//Locutor7
med[6][0]=0.0738736;
      var[6][0]=0.0001335649;
med[6][1]=0.1223812;
      var[6][1]=0.0020568394;
med[6][2]=0.1743744;
      var[6][2]=0.0064107065;
med[6][3]=0.27654;
      var[6][3]=0.0033580645;
med[6][4]=0.3688726;
      var[6][4]=0.0066818005;
```

```
med[6][5]=0.4916048;
      var[6][5]=0.0003065515;
med[6][6]=0.536343;
      var[6][6]=0.0006726202;
med[6][7]=0.6279906;
      var[6][7]=0.0043935806;
med[6][8]=0.7245644;
      var[6][8]=0.0004339719;
med[6][9]=0.1690742;
      var[6][9]=0.0001567448;
med[6][10]=0.2660666;
      var[6][10]=0.0000843135233;
med[6][11]=0.3151616;
      var[6][11]=0.0001116141;
med[6][12]=0.4019954;
      var[6][12]=0.0003645805;
med[6][13]=0.44395;
      var[6][13]=0.0001240078;
med[6][14]=0.5124272;
      var[6][14]=0.0014602048;
med[6][15]=0.582866;
      var[6][15]=0.0002006764;
med[6][16]=0.6121572;
      var[6][16]=0.0002384124;
med[6][17]=0.6554304;
      var[6][17]=0.0003281043;

//Locutor8
med[7][0]=0.0743818;
      var[7][0]=0.0000705664762;
med[7][1]=0.11815;
      var[7][1]=0.0014193698;
med[7][2]=0.1925714;
      var[7][2]=0.0027166004;
med[7][3]=0.2841576;
      var[7][3]=0.001528216;
med[7][4]=0.382108;
      var[7][4]=0.0029651002;
med[7][5]=0.4777084;
      var[7][5]=0.0084104315;
med[7][6]=0.578751;
      var[7][6]=0.0045497289;
med[7][7]=0.6767196;
      var[7][7]=0.0001837122;
med[7][8]=0.740431;
      var[7][8]=0.0002403254;
med[7][9]=0.1560502;
      var[7][9]=0.000262768;
med[7][10]=0.250813;
      var[7][10]=0.0002427824;
med[7][11]=0.2830892;
      var[7][11]=0.0001900234;
med[7][12]=0.3166392;
      var[7][12]=0.0001723706;
med[7][13]=0.4196226;
      var[7][13]=0.0002969761;
med[7][14]=0.5003594;
      var[7][14]=0.0019634732;
med[7][15]=0.5800978;
      var[7][15]=0.0006255534;
med[7][16]=0.6242396;
      var[7][16]=0.000557931;
med[7][17]=0.7711972;
      var[7][17]=0.0004691144;

//Locutor9
med[8][0]=0.0818424;
      var[8][0]=0.0001296008;
med[8][1]=0.1453716;
      var[8][1]=0.000593825;
med[8][2]=0.227797;
      var[8][2]=0.0006895809;
med[8][3]=0.3461694;
      var[8][3]=0.0007887066;
med[8][4]=0.4165546;
      var[8][4]=0.0014101945;
med[8][5]=0.5186454;
      var[8][5]=0.0007656814;
med[8][6]=0.6051666;
      var[8][6]=0.002360422;
med[8][7]=0.711241;
      var[8][7]=0.0008659038;
med[8][8]=0.8067414;
      var[8][8]=0.0012957813;
med[8][9]=0.1909596;
      var[8][9]=0.000512249;
med[8][10]=0.2536742;
      var[8][10]=0.0001678398;
med[8][11]=0.3186168;
      var[8][11]=0.0004605661;
med[8][12]=0.4265618;
      var[8][12]=0.0005456592;
med[8][13]=0.4605822;
      var[8][13]=0.0004697244;
med[8][14]=0.5317546;
      var[8][14]=0.0019377284;
med[8][15]=0.6051956;
      var[8][15]=0.0004986959;
med[8][16]=0.6421846;
      var[8][16]=0.0003934554;
med[8][17]=0.7753456;
      var[8][17]=0.0007381876;

//Locutor10
med[9][0]=0.0585858;
      var[9][0]=0.000101062;
med[9][1]=0.07073;
      var[9][1]=0.0001310422;
med[9][2]=0.085057;
      var[9][2]=0.0001245454;
med[9][3]=0.1143994;
      var[9][3]=0.000115927;
med[9][4]=0.2447346;
      var[9][4]=0.0025631442;
med[9][5]=0.3528542;
      var[9][5]=0.0040220887;
med[9][6]=0.4692018;
      var[9][6]=0.0033920877;
med[9][7]=0.6181008;
      var[9][7]=0.0037310891;
med[9][8]=0.6921752;
      var[9][8]=0.0001947875;
med[9][9]=0.1557972;
      var[9][9]=0.0001398264;
med[9][10]=0.2507398;
      var[9][10]=0.0002690751;
med[9][11]=0.3015448;
      var[9][11]=0.0003126359;
```

```
med[9][12]=0.387581;
      var[9][12]=0.0005839477;
med[9][13]=0.4310446;
      var[9][13]=0.0003214747;
med[9][14]=0.5233354;
      var[9][14]=0.0006501357;
med[9][15]=0.5649408;
      var[9][15]=0.0003390023;
med[9][16]=0.5982476;
      var[9][16]=0.0003558201;
med[9][17]=0.7178792;
      var[9][17]=0.0005703279;

//Locutor11
med[10][0]=0.0700614;
      var[10][0]=0.0000733797143;
med[10][1]=0.1014604;
      var[10][1]=0.0000310522273;
med[10][2]=0.1533486;
      var[10][2]=0.0016602049;
med[10][3]=0.2233822;
      var[10][3]=0.0017577359;
med[10][4]=0.3108018;
      var[10][4]=0.0041395563;
med[10][5]=0.389738;
      var[10][5]=0.0009143339;
med[10][6]=0.4652604;
      var[10][6]=0.0027378319;
med[10][7]=0.5964206;
      var[10][7]=0.0057661625;
med[10][8]=0.734967;
      var[10][8]=0.000403716;
med[10][9]=0.1964364;
      var[10][9]=0.0003967443;
med[10][10]=0.2708612;
      var[10][10]=0.0001570076;
med[10][11]=0.319194;
      var[10][11]=0.0002180351;
med[10][12]=0.3854022;
      var[10][12]=0.0010846449;
med[10][13]=0.4574316;
      var[10][13]=0.000887337;
med[10][14]=0.506861;
      var[10][14]=0.001713856;
med[10][15]=0.5967414;
      var[10][15]=0.0008182378;
med[10][16]=0.634973;
      var[10][16]=0.001014534;
med[10][17]=0.7125286;
      var[10][17]=0.0004665493;

//Locutor12
med[11][0]=0.1092718;
      var[11][0]=0.0044541463;
med[11][1]=0.1972286;
      var[11][1]=0.006350034;
med[11][2]=0.2595168;
      var[11][2]=0.0049370036;
med[11][3]=0.349759;
      var[11][3]=0.0056348257;
med[11][4]=0.3860972;
      var[11][4]=0.0038440031;
med[11][5]=0.4687198;
      var[11][5]=0.0028911516;
med[11][6]=0.5731024;
      var[11][6]=0.0041980086;
med[11][7]=0.6357846;
      var[11][7]=0.0006473051;
med[11][8]=0.6788844;
      var[11][8]=0.0008977024;
med[11][9]=0.1421684;
      var[11][9]=0.0006367437;
med[11][10]=0.2183596;
      var[11][10]=0.0002658034;
med[11][11]=0.2707632;
      var[11][11]=0.0003640667;
med[11][12]=0.3585938;
      var[11][12]=0.0011551662;
med[11][13]=0.403273;
      var[11][13]=0.0001442423;
med[11][14]=0.4865612;
      var[11][14]=0.0012657802;
med[11][15]=0.5304188;
      var[11][15]=0.000127073;
med[11][16]=0.5592556;
      var[11][16]=0.00013427;
med[11][17]=0.6367874;
      var[11][17]=0.0027355498;

do{
//Entrada do sinal a ser analisado
      for(i=0;i<18;i++){
            printf("Entre com a posicao
%i do vetor a ser analisado:",i+1);
            cin >> sinal[i];
      }

//Calculo dos modelos
      conv=0;
      indice=-1;
      for(i=0;i<12;i++){
            modelo[i]=exp(-
(dist2(sinal,med,i))/(2*modulo(var,i))
);
            if(modelo[i]>conv){
                  conv=modelo[i];
                  indice=i;
            }
      }

indice=indice+1;

//Apresentacao do resultado
      printf("Converge com o locutor
%i, com resultado de
%f!",indice,conv);

      printf("Para continuar digite um
inteiro diferente de zero");
      scanf("%i",&j);
}while(j);

return 0;
}
```